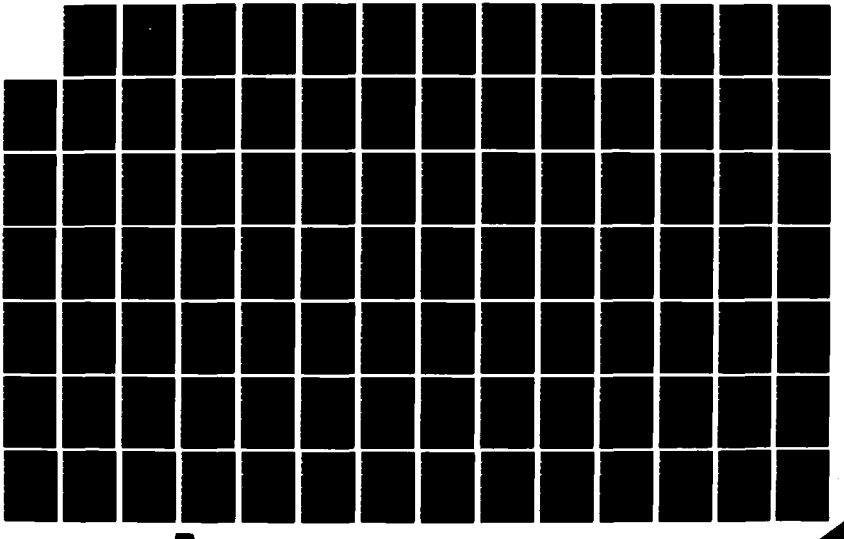
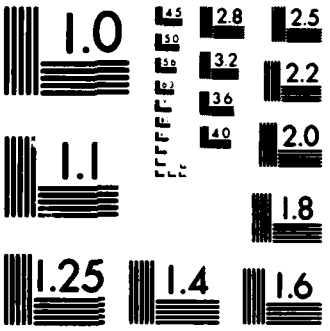
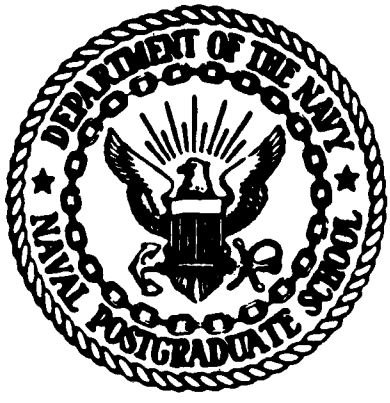




(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

TACTICAL USE OF DIGITIZED MAPS

by

Steven James Gaffney
and
James Earl Daly

September 1987

Thesis Advisor: Norman R. Lyons

87 11

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is Unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 54	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
TACTICAL USE OF DIGITIZED MAPS (u)

12 PERSONAL AUTHOR(S)
Gaffney, Steven James and Daly, James Earl

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 September	118

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Digitized mapping system; Tactical Terrain Analysis Data Base; Computer Mapping; Digitized Maps; Digital Mapping System

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this study was to determine the problems and possibilities of developing a digitized mapping system for the ground tactical commander. The specific tactical units targeted were the Marine Infantry regiment and below. Particular emphasis was placed on the microcomputer as the implementation hardware. A review of the Defense Mapping Agency (DMA) databases was conducted and related Department of Defense programs were studied. To develop insights and to provide a graphical tool for understanding the problems of a digitized mapping system, a Prototype was developed. It was determined that current microcomputer technology and DMA data bases provide the capability to develop and field an adequate, if not optimum, digitized mapping system.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Norman R. Lyons	22b TELEPHONE (Include Area Code) (408) 646-2666 22c OFFICE SYMBOL Code 54Lb

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited.

Tactical use of Digitized Maps

by

Steven James Gaffney
Major, United States Marine Corps
B.S., United States Naval Academy, 19--
M.S.B.A., Boston University, 1985

and

James Earl Daly
Captain, United States Marine Corps
B.S. Mgt., University of Utah, 1981
B.S. Fin., University of Utah, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

Authors: _____
Steven James Gaffney

_____
James Earl Daly

Approved by: _____
Norman R. Lyons, Thesis Advisor

_____
Joseph S. Stewart II, Second Reader

_____
Willis R. Greer, Jr. Chairman, Department of Administrative Science

_____
Kneale T. Marshall, Dean of Information and Policy Sciences

ABSTRACT

The purpose of this study was to determine the problems and possibilities of developing a digitized mapping system for the ground tactical commander. The specific tactical units targeted were the Marine Infantry regiment and below. Particular emphasis was placed on the microcomputer as the implementation hardware. A review of the Defense Mapping Agency (DMA) databases was conducted and related Department of Defense programs were studied. To develop insights and to provide a graphical tool for understanding the problems of a digitized mapping system, a Prototype was developed. It was determined that current microcomputer technology and DMA data bases provide the capability to develop and field an adequate, if not optimum, digitized mapping system.

TABLE OF CONTENTS

I. PROBLEM SUMMARY

A. BACKGROUND

The modern battlefield is a highly mobile and fluid environment requiring the Tactical Commander to be acutely aware of terrain. He needs to "get the lay of the land" or to gather data without having to be there, and he must risk equipment and the lives of his men on this information. A system of paper maps is presently in use which is adequate for the job; however they are sometimes slow and tedious to use. Maps are annotated with pre-planned targets, color-coded terrain features, coordination lines, routes, check-points, positions and more. Data is extracted and decisions are made based on its analysis. Additionally, many of the preparations by a commander rely on the mental pictures of the terrain built upon the data derived from the map. Potential for error exists for all these operations. Because a map was misread or the situation didn't allow sufficient time to extract the proper information, an untold number of men have died.

Digital databases which are being fielded now by the Defense Mapping Agency (DMA), for the first time provide a source of data which represent the information contained in a paper map. With the availability of these new databases and, in light of the state of current and future technology, there exists the possibility that maps can be computerized for field use. The focus of this study was to determine if there is now the technical feasibility to develop a worthwhile digitized mapping system for ground tactical commanders.

B. SCOPE

1. Tactical User

Computers in the field produce their own set of problems. Power requirements, transportability, survivability and electronic radiation must be of concern. The capabilities to use or support sophisticated equipment vary greatly from the rear areas to the forward edge of the battlefield. No one solution is appropriate to both environments. Accordingly, to insure a consistent focus, this study has been limited to the Marine infantry battalion or regimental level.

2. Hardware

A machine used in the field by a tactical commander must be extremely mobile. A Marine Infantry Battalion or Regiment has finite, and extremely limited, motor transportation and power generation assets available in its Table Of Equipment (T/E), and required movement is commonly over very rough terrain. The requirement for special facilities (buildings, vehicles, or special generators) would make the support of a computer in the field implausible, as would the requirement for special support personnel or significant special training. The facilities available to a battalion or regimental commander are only capable of supporting a machine not requiring dedicated environmental support (i.e. microcomputer or minicomputer).

In the final analysis, there is a trade-off between environmental constraints and system capabilities. There has always been a demand for equipment with fewer and fewer environmental needs as you approach the battlefield. Modern warfare, with high mobility and sophisticated targeting and intelligence-gathering capabilities has in no way diminished this. Computing technology improvements have at the same time increased the abilities of smaller, less environmentally demanding systems. Any technological

solution that significantly increases the logistical burden, increases vulnerability, or reduces the mobility of the fighting Marine will be politically unacceptable. Accordingly, the focus of this study is on the technological capabilities and constraints of the microcomputer. No attempt has been made to assess the political or economic feasibility of the microcomputer or the feasibility of minicomputers or mainframes.

3. Data

Probably the most critical issue of this study was the availability of usable data. The creation of an entirely new database to support tactical digitized mapping was considered to be impractical. The DMA has the mission of "...producing and distributing to the Joint Chiefs of Staff, Unified and Specified Commands, Military Departments and other Department of Defense users complete, credible, and effective mapping, charting and geodetic products." ("Defense Mapping Agency: Digitizing the Future", October 1986) They currently have seventeen digitized databases and are constantly researching others. Given the requirement for a reliable and complete data source and the dedication of the DMA to that task, it was determined that only data products of the DMA would be considered for use.

Because of the large quantities of data contained on a map, digitized maps from DMA are presently available only on nine track tape. These tapes are not the ideal media for microcomputers and must be converted to a usable format. The effects of this conversion process were an important issue but the exact technical procedure was considered to be outside the scope of this study.

C. METHOD OF STUDY

The study was conducted in four parts. First, the available databases of digitized mapping data were reviewed.

The appropriate database was identified and its strengths and weaknesses were determined. The needed data modifications for field use on a microcomputer were analyzed and recommended. Next, Department of Defense initiatives in the area of digitized mapping were reviewed to determine what advances, relevant to this study, had been made. A prototype was developed to expand the researchers' understanding of programing, data and hardware problems and limitations. The prototype duplicated some areas previously addressed by other programs but also addressed new data concerns. Finally, possible hardware solutions, to problems uncovered in the study, were put forward.

II. DATA

A. DATABASE AVAILABILITY

The researchers operated under the premise that it is not viable to automate tactical maps unless existing databases provide all required information. Creating a new "world-wide" database would be a monumental effort and is well beyond the reasonable level of effort and expense for the sole purpose of the tactical automation of the present mapping system. Furthermore, three forms of relevant map data are available through the DMA. The following three available databases were analyzed to determine if one or more of them were adequate for the envisioned project.

1. Digital Terrain Elevation Data

Digital Terrain Elevation Data (DTED) has been used for several existing computer mapping projects (e.g. Terranal, MicroDem, and Microfix)[1]. DTED however, gives only the elevation data for a given piece of terrain one degree of latitude by one degree of longitude (1° X 1°), which in its complete form requires two megabytes of media. It is stored as a binary stream in 1° X 1° cells on 9 track 1600 bpi tape. DTED is very complete in its representation and provides adequate information for the elevation or contour display, but lacks the feature data that is key to this thesis. It would need to be converted to a usable format for our purposes.

2. Digital Features Analysis Data

Digital Features Analysis Data (DFAD) is detailed in its feature representation of terrain, however, it does not contain the necessary elevation data. It, like the DTED, is stored as a binary stream in 1° X 1° cells on 9 track 1600

---

[1]. See Appendix A.

bpi tape, yet is in an entirely different format. This would require another translation utility program to make it usable on the target computer. The DFAD and the DTED would then require combining into one format, if possible, to be of use in a tactical mapping system.

3. Tactical Terrain Analysis Data Base

The Tactical Terrain Analysis Data Base (TTADB) is a new storage format in the implementation phase of development by DMA. It provides an integrated source of both elevation and features data in a common format, and will be fully on line in the early 1990's. It is designed to conform to a 1:50,000 map presentation format. The storage is in a binary stream of terrain points (at 12.5 meters intervals) on 9 track 1600 bpi tape. A translation utility program has already been developed for another Marine Corps project (Amphibious Objective Area Land Management System) by the Naval Civil Engineering Laboratory (NCEL) of Port Hueneme, California.[2] This program could be converted or adapted to produce the exact output required for a tactical digital mapping system.

4. Selection

The availability of these three databases from DMA, with world-wide coverage, provides for a practical database source. A combination of the DTED and the DFAD may be possible, but could be difficult to implement and may introduce unacceptable compromises of accuracy or detail. The TTADB offers a consistent data package without the need for integration. Accordingly, the TTADB was chosen as the practical database for further study.

B. DATABASE DESCRIPTION

The Fort Lewis-Yakima Training Area prototype of the TTADB was used as the data model for this study. This data

---

[2]. See Appendix A.

is not in the final form of the TTADB, but the prototype model should allow for general concept illustration. The Fort Lewis-Yakima Training Area prototype TTADB data is made up of 199 bit terrain point records which describe the following features:[3]

-- Surface Configuration
-- Vegetation
-- Surface Material
-- Surface Drainage
-- Transportation
-- Obstacles

The full proposed TTADB is made up of 224 bits per terrain point record, and includes the following additional feature descriptions:

-- Aerial Obstructions
-- Special Features/Product Synthesis
-- Text Data

Each data point of TTADB is 12.5 meters apart. This results in each square kilometer being represented by a grid 80 X 80 points, which is 6400 points of 25 to 28 bytes of 8 bits each. Thus each square kilometer could take as much as 179,000 bytes. To represent even a small map of 400 square kilometers would require over 70 Megabytes of storage. This is a prohibitively large data base for both storage and manipulation in a general-purpose microcomputer environment of 1987.

C. DATABASE CONVERSION

As previously noted, the TTADB as provided by DMA is on an inappropriate data media for micro-computing; 9 track tape, and is extremely bulky. To make the data useful, it needs to be converted to a suitable form and a reduced

---

[3]. See Appendix B.

volume. In doing this, it is important that no significant details or information be lost.

1. Point Reduction

The first step towards that goal is to reduce the number of data points. A tremendous amount of space is used in the TTADB to provide a significant degree of accuracy. This is of grave concern to some parties who would use this data base. In fact, on several occasions Captain M. R. Reading of the Defense Mapping School at Fort Belvior, Virginia warned the researchers of the accuracy of the TTADB data being only to plus or minus 37.5 meters.[4] "Topographic products, such as maps, are produced to varying degrees of accuracy, depending on the source material availability, survey control, and the scale at which the product is published."(FM 21-32, 1979, p. 1-9) Accuracy of this degree is of primary concern to engineers, geologists, and topographers; this is important to a much lesser degree to tactical ground units. It was determined that the 12.5 meter resolution of the data points was not necessary to support this project. The use of only one in four of the data points would suffice. This would place each point 50 meters from the next and no piece of terrain would lie over 35 meters from a data point. Any feature that is represented by one of the ignored data points should be represented in the nearest point selected for retention. As long as all appropriate data from the three deleted points can be transferred to the remaining point, no information should be lost. The accuracy of the converted data base would remain largely within the 37.5 meters of accuracy cited by Captain Reading. The most significant error could be 75 meters. That is, a point 37.5 meters off, being 12.5 meters into the preceding or following four point grouping,

---

4. Telephone conversations between Captain Daly and Captain Reading (an instructor at the Defense Mapping School), 10 February 1987, and 17 April 1987.

results in the data being displaced 37.5 meters further, for a total inaccuracy of 75 meters. Considering that this is the worst case, the reduction seems reasonable.

Of more concern than the possible inaccuracies generated, is the possible loss of feature data. The three deleted points cannot merely be deleted. Any appropriate data must be passed to the nearest consolidated data point. Any conflicts , e.g. one point has swamp another has forest for a vegetation code, must be resolved based on a sound, well considered, hierarchy of importance. Fortunately there are seldom cases where four different points 12.5 meters apart each have different characteristics, each of which are tactically significant.

2. Field Reductions

Having consolidated the points, the next issue is the amount of relevant data contained in the data base. All of the data in TTADB is of value to someone and there are probably cases that can be imagined where most of the data could be used for some tactical purpose. Realistically, the majority of information is of little or no use to the ground tactical commander. When required, much of the data provided could and would be confirmed by individuals on the scene. Certainly, too much of the information is of a perishable nature or seems to be difficult to obtain. In any case the data would possibly be misleading or unavailable.

Accordingly, the number of data fields was reduced to more realistically reflect the significant needs of the tactical users. While the decisions made are arguable, they are at least logical. An in-depth review of the data required is beyond the scope of this study. Decisions were made as to what to cut and what to retain on the basis of what an individual expects to see on a paper map and what information would and should be relied on after the data is

aged. Extras and time sensitive data were deleted. These reductions resulted in a more streamlined, concise, and understandable data base.[5]

3. Field Consolidations

To reduce the data fields requirements even more, data fields which provided qualifiers were added to the fields which they qualified. This adds an additional requirement that the data fields be integrated with prioritization of which data is represented in cases of conflict. In most cases the relationships lend themselves to easy prioritization. The surface drainage data was appended to the vegetation overlay. The highways and roads data fields were added to the transportation overlay as was the transportation qualifier field. This final cut reduces each point's storage requirement to less than four bytes. The 400 square kilometer map which was impossibly large in the original form, can now be stored on two standard 360 KB, 5 1/4 inch diskettes with room to spare.

4. Prototype Format

With the data minimized, consideration has to be given to a usable format. The first decision was to change to a byte ASCII format, versus a binary bit format. This approach offers three advantages. First, each edited byte field has the potential for future expansion. That is, representing the vegetation currently requires only 5 bits which allows only 32 bit combinations. The use of a full byte allows 256 combinations. Second, bytes and ASCII characters can be dealt with more straight forwardly in the DOS environment. Finally, and most importantly, the ASCII byte format can be read easily by humans. Information can be extracted from the data by looking at it, whereas binary is all but incomprehensible.

---

[5]. See Appendix C.

Another concern is for expansion from the minimum data base to include fields discarded in this review, but required by others. Also there is a need to include space for locally applied information (ie. targets, unit locations, bunkers, etc.). To accommodate this concern, two blank byte fields were added. The final prototype data format is as follows:

TABLE 1. TDMS PROTOTYPE DATA FORMAT

Byte	Purpose	Remarks
1-4	Elevation	+9999 to -999 meters
5	Transportation overlay	Combines Type, Qualifier and Highways / Roads Type fields
6	Vegetation overlay	Combines Vegetation and Surface Drainage Type fields
7-8	Expansion/local use	Not used

The prototype format provides a reasonable basis for evaluation. While it does not include all data of the original TTADB, it includes its significant detail and data fields. The byte versus bit format and the addition of two bytes allows for moderate expansion. If more detail is required, going to a bit format within the 8 byte size could accommodate a 60% increase. With 8 bytes a 400 square kilometer area can be stored on one 3 1/2 inch 1.44 megabyte diskette. A 20 megabyte hard disk would store well over 5000 square kilometers.

5. Conversion Summary

Table number 2 illustrates the data size relationships for the above described reduction efforts:

TABLE 2. TDMS DATA STORAGE REQUIREMENTS

Points per sq k	Size	Max total bytes	400sq k bytes
Initial 80x80=6400	25 to 28 bytes	179,200	71,680,000
AFTER Reducing Points 20x20=400	25 to 28 bytes	11,200	4,480,000
AFTER Reducing Fields 20x20=400	33 bits/5 bytes	2000	800,000
AFTER Consolidating Fields 20x20=400	25 bits/4 bytes	1600	640,000
Prototype Format 20x20=400	8 bytes	3400	1,360,000

D. TEST DATA GENERATION

The data for the thesis was manually generated in the prototype format to emulate TTADB data after reduction and reformatting as discussed above. Actual data was not utilized for three reasons. First, the creation of a mainframe conversion program is outside the scope and resource limitations of this study. Second, the relatively small amount of data required (8 square kilometers) does not justify the effort. Lastly, the creation and use of actual data would not provide any additional insight and would be disappointingly lacking in mixes of features desired to adequately test the prototype mapping display. Therefore, data was designed in the TTADB format, with very dense terrain features to test the display and prototype program.

E. DATA CONCERNS

Having reviewed databases and created test data, several broad general concerns, outside the scope of study, developed. The TTADB does not contain information for

labels such as town names or road numbers. The proposed "text data" fields may include some information, but without greatly increasing the record size little information could be passed through that means. The next concern is that of users perception of data reliability. The TTADB has not had years of field use nor has it even been implemented. Conventional maps are trusted because they are a known resource and can be judged by their overall appearance of quality. Digitized maps are a new, untested element and, since they reside as electronic bits and bytes, cannot be touched and studied. This makes it difficult for the user to develop confidence in the product regardless of how good it may be. Another concern is for the sharing of information between data points. Each point is concerned only with the data at that location. Often features only gain significance as a group of points. It is difficult to envision that the complete image of a feature, taken apart piece by piece, can always be reassembled without distortion. These concerns are outside of the scope of this study and are offered for consideration only.

III. RELATED DEPARTMENT OF DEFENSE PROGRAMS

All programs of a similar nature were studied to determine what lessons could be learned. The following programs were discovered to use digitized data on a computer system for terrain analysis.

A. PROGRAMS

1. AOALMS (Amphibious Objective Area Land Management System)

The AOALMS is a prototype that was designed by the Naval Civil Engineering Laboratory for the Marine Corps, and written in Janus Ada, to assist in rapid planning of horizontal construction projects before arrival in the Amphibious Objective Area (AOA). It is for use by the Landing Force Commander and the Engineer Support Battalion for the planning and construction of facilities worldwide.

The system consists of an MS-DOS microcomputer and software programs that aid in: (1) examining site conditions and identifying promising locations for construction, (2) designing the facilities and calculating earthwork requirements, (3) determining the numbers and types of construction equipment needed to meet completion schedules, and (4) in scheduling the construction effort to ensure optimum usage of construction equipment. The primary function being considered (and the only one presently addressed by the program) is construction of expeditionary airfields. Funding for this project was discontinued before development could be completed.[6]

---

[6]. See Appendix A.

2. VISTA (VISION, INTERVISIBILITY, SURROGATE TRAVEL, TERRAIN ANALYSIS)

VISTA is an interactive graphic terrain analysis program developed by the Armored Systems Division, US Army TRADOC Systems Analysis Activity (TRASANA), for detailed study of terrain. It was written in Fortran 77 and is implemented on Digital's VAX 11-780 computer system using high resolution, color, graphics terminals to display and analyze terrain data in a variety of ways. The supporting data base was constructed specifically for the project based on DMA's TTADB, and consists of elevations, surface features, roads, rivers, hydrography, and obstacles. VISTA currently operates on a 12x12 kilometer area at 100-meter resolution. The terrain data can be displayed in map form as shaded relief, color contours, or regular contours. Three modes are used to examine the map display: Line-Of-Sight (LOS), Perspective Viewing, and Surrogate Travel. LOS displays the view as observed from different points on the map. Perspective Viewing allows the user to position an observer at any point on or outside the terrain to obtain a 3-D view of the terrain and surrounding features as they would appear from that point. Surrogate Travel presents an animated 3-D perspective view of travel along defined surface or aerial routes within the displayed terrain area. The visual representation of built-up areas and vehicular movement are the strengths of this program.[7]

3. MICRODEM

MicroDEM was developed at the U. S. Military Academy to support the Army's needs for computer assisted terrain analysis. It manipulates large digital terrain models, developed from DMA's Digital Terrain Elevation Data base, to produce; (1) color tinted contour maps, (2) slope maps, (3) masked topographic profiles, and (4) 3D oblique views of terrain. The program will run on MS-DOS computers equipped

---

[7]. See Appendix A.

with a CGA monitor, at least one disk drive and 256 Kbytes of memory. It was written in Borland International's Turbo Pascal (ver. 3.0, 1985). The MicroDEM program most closely matched the concepts and goals of this study, therefore is was analyzed in some depth, as described below.[8]

B. MICRODEM/TERRANAL

1. Background

TERRANAL is a program developed by the Computer Graphics Laboratory of the United States Military Academy. The originator of the project was then Captain Peter L. Guth, an instructor in the Geography department. He was assisted by Captain Eugene K. Ressler and others before he resigned from the Army to take a position in the Geology Department at University of Nevada at Las Vegas. Captain Ressler took over cognizance of the project, but Major Guth (USAR) continues to work on the program to "civilianize" it. It was later released as MicroDEM, which had a few refinements over the original.

TERRANAL was conceived and designed to: (1) provide the Army with a testbed to develop new tactically oriented computer mapping systems, (2) to put digital mapping and terrain analysis in the hands of all professional terrain analysts and topographers, and (3) to provide prototype systems that could teach field users about digital terrain mapping.

A derivative parallel version of the program called MICROFIX operates on an enhanced Apple II+ (the AN/UYK-71).

2. Accomplishments

The program was written in Turbo Pascal for an MS-DOS microcomputer with a CGA display, which proves that a microcomputer environment can process and display the digitized data. MicroDEM is adequate in breadth and depth

---

[8]. Also see Appendix A.

to prove that a microcomputer can be used to generate and display maps which provide some useful planning operations, as illustrated by the following list of functions:

-- Color tinted elevation maps, with seven user defined categories, at any desired scale;

-- Slope maps, with four user defined categories, at any desired scale;

-- Contour maps, at any desired scale and contour interval, and masked area plots showing terrain hidden from an observer;

-- Topographic profiles between any two desired points, at any desired scale and vertical exaggeration;

-- Oblique three dimensional views with any desired orientation;

The capabilities provided are significant, and may become part of any tactical digitized mapping system that is implemented. However, the critical features displays are not available and some of the functions are not fully developed.

The ability to use existing and available DMA data is also proved by MicroDEM. By using the DTED DMA database, the ability of a microcomputer to provide several graphic displays of terrain elevation characteristics is proven. Storing and transporting data for the mapping system is a critical question that was also addressed. MicroDEM proved that the data base can be broken into units small enough to be carried on floppy disks, and still provide adequate information to build a usable map display.

3. Limitations

Many of the functions provided would be useful and valuable to an infantry officer, yet MicroDEM uses only elevation data. Failing to use terrain features limits its value.

It was apparently developed with topographic studies in mind, not for the tactical commander's needs. As such, it provides good information, but lacks some of the

characteristics (most relating to data manipulation regarding terrain features) that would make it most valuable in the field.

The area displayed by the program is a grid 15' by 15', which is an adequate size for tactical users. However, the DTED database provides only 90% accuracy of points 60 - 120 meters apart East/ West and separated by 92 meters North/ South. Additional inaccuracies are probably introduced in conversion to 64,000 points (320 by 200 pixels) for display as individual pixels on the CGA display. This resolution may be adequate to represent the single dimension of elevation data, but it is not possible to display features as individual pixels. The system is also slow in that, after the calculation that determines its position in relation to the data point it represents has been done, each pixel is written to the screen individually.

C. EVALUATION

This research is an effort to determine if microcomputers can display and use map data necessary for the terrain analysis that would be performed by a tactical commander. All of the projects studied are related to this thesis, yet the relationship is remote in each case.

AOALMS used the TTADB to display a computer-generated map, but does not attempt to use the features data available. Vista is based on the TTADB and displays features, but is a mainframe computer system which is capable of significantly greater tasks than a micro is capable of handling. MicroDEM has made significant strides toward the functional use of microcomputer map displays but is missing the features and resolution that are critical to a tactical planner or field commander.

These programs have helped validate some of the concepts that this study is aimed to prove; however, many questions

are not adequately addressed. Can current technology support the resolution to display features in addition to elevation? Can the microcomputer micro-processor provide the speed and power to manipulate larger data files to support large maps? Is the storage media capable of survival in a field environment, and providing storage of the large amounts of data required?

IV. TACTICAL DIGITAL MAPPING SYSTEM PROTOTYPE

To develop an understanding and to provide a graphical tool reviewing the problems of a digitized mapping system, the Tactical Digital Mapping System Prototype (TDMS) was developed. The system was not intended to be a fully developed system, but merely an exploration of concepts. The MicroDEM system provides an excellent example of the possibilities of microcomputer based digitized mapping. It did not, however, show an operationally oriented environment nor did it include the features data required of a fully functional system. It was not the intention of the researchers to duplicate the efforts of the MicroDEM system but to develop three concept areas. First it was desired to explore the possibilities of the outward system appearance to the user. Second it served as a vehicle to develop an appreciation of the display requirements and limitations. Finally, it forced an applied study of the data requirements discussed in Chapter III as well as providing an insight to undeveloped data requirements. The complete prototype program listing is provided.[9]

A. DESCRIPTION

The Tactical Digitized Mapping System prototype was written in the 'C' language. The program consists of three major areas; a program shell divided into functional areas, a portion that consolidates the user requested data, and a display section. In operation, the user is guided by menus to input grid coordinates and then a map is produced and displayed. The display is a full color map of five square kilometers with features and elevation represented. The

---

[9]. See Appendix G.

system was designed to work on an IBM PC or compatible with an EGA display. All files for the program must be located on the same drive to operate. The prototype is not intended to be fully functional but merely to illustrate the look and feel of the implemented system.

B. DATA

The prototype served as an instrument for determining the data format requirements. In addition to the considerations of Chapter II, the structural layout of the map data files needed to be determined. Each one kilometer block of data was placed in an individual file. The filename indicated the grid coordinate of the data. For example, D6008.DAT indicated a data file of the one kilometer square with the lower left hand grid coordinate 6008. To assemble larger map units the files were consolidated. The use of the grid coordinate in the filename allowed the mathematical manipulation of filenames to determine or isolate desired files. To create a four square kilometer map with the grid coordinate 6008 at the lower left hand corner, one is added to 6008 to determine the upper left hand corner, 101 is added for the upper right and 100 for the lower right. An 'M' replaced the 'D' to designate a composite map data file, M6008.DAT.

While combining one square kilometer data files is a practical approach to manipulating data files, it is also slow. To consolidate an eight square kilometer file, with an AT accessing files on a RAM disk, required six seconds. Even with an optimized routine, which this was not, the consolidation operation for a 100 square kilometer map may be slower than desired. One alternative may be to build bigger basic building blocks. The difficulty with this approach is that all operations will now have to deal with the larger blocks, not just the consolidating routine. The

larger blocks in some cases will require more data to be transferred than the smaller blocks. A more promising approach may be to use the data directly, when appropriate, without consolidating files. A faster central processing unit (CPU) and making full use of techniques for fast memory transfers may reduce this problem to an acceptable level.[10]

The one kilometer data blocks provide a logical base for data searches and extractions. By using a one kilometer block a logical addressing system is easily programmed. Searches and extractions can be given in terms of grid addresses. The Military Grid Reference System equally divides a grid coordinate between the kilometers east of an ordinate and the kilometers north of that ordinate. That is, the coordinate 614825 is 61.4 kilometers east of the 000000 coordinate and 82.5 north. To exploit this characteristic, a data block should be a square oriented on an evenly matched coordinate pair. For example, the coordinate pair 68 would represent a square with sides running east from 60 to 69.9 and north from 80 to 89.9. The next choice would be the four digit coordinate pair 6182 which represent a square kilometer. Using this method, orderly addressing algorithms can be developed and user input does not have to be translated into an entirely different addressing scheme. As the two digit coordinate pair results in a 100 square kilometer block and the six digit pair results in a .01 kilometer square, the four digit, one kilometer square is the logical file block size.

C. OPERATION

1. General

Any system for wide general military use, must be suited for the computer illiterate with a high school diploma. Rapid turnover of personnel puts emphasis on

---

[10]. See Chapter V.

minimizing training requirements. If the system cannot deliver on the unstated pledge of faster results and ease of use, there will be little acceptance. While the system may have many capabilities, each individual user should be able to access and run his application without undue stress and with a minimum of instruction. There is nothing to prohibit the development of a fully functional system with a poor user interface, but addressing these issues is essential to providing a quality product. The prototype environmental shell was conceived to have a targeted user environment, with no requirement for extensive computer knowledge on the part of the routine user. The common practice of providing an error checked, menu driven system was followed. When the targeted user is in the system he should have no concerns about computer system issues. As long as the user can read and knows his job, the system must not provide him with technical or impossible demands. The prototype system is able to do this, but a full system may have difficulties. The prototype was designed for one computer environment only, had no conflicting hardware requirements (e.g. mouse or keyboard, monochrome or color), and dealt with relatively simple operations.

By way of contrast, the MicroDEM system provides a general system which can be configured in a variety of ways. The numerous capabilities of the system require the user to establish, or maintain, the proper environment, steer through general system menus and know and recall specific commands relating to various program portions. The researchers experienced problems in doing some routine tasks and were thrown by several environmental set up issues. Had the MicroDEM program been written for specific equipment, tasks, and users with ease of use as a priority none of these problems would have developed. If the user only had to select from menus of items he is familiar with, and

commands and instructions were available and consistent, the user could not make mistakes. This is not to criticize the system, which is not a finished product or an application for a specific use. The point is, that had simplicity of use been a priority in the development of MicroDBM, the same system which is troublesome for a computer literate could be quite friendly for the novice user.

2. Title Screen

When the system is started the TDMS title screen is displayed. After a timed delay or a key press, the screen is cleared and the main menu is displayed. In the full implementation, this screen could be used to display general information such as security requirements or operating restrictions.

3. Main Menu

The main menu displays five choices representing four of the functional areas of a battalion or regimental staff and an exit option. By selecting one of these the user is put into a subsystem tailored to that functional area or is exited from the system.

In developing the system shell, the concept of how the system would be used operationally, while not entirely within the scope of the study, had to be addressed. The targeted users are primarily the commanders and staff of Marine battalions, regiments, and divisions. The principle tactical map data processors of these organizations are the operations officer, intelligence officer, the fire support coordination center staff and the logistics officer. Each of these have unique but similar requirements for data manipulation and the output of one may be the input to another. It was therefore logical that the system be divided into modules for each functional area. Since each functional module shares many of the same requirements, it is logical that they share common subordinate modules. For

the prototype, the system was completely integrated. The issue that must be resolved in an implemented system is; how far should integration go? Should four distinct functional area systems be developed or can one system handle all the requirements without growing too large? In a shared environment do some users need to be limited in their access to system features?

4. Functional Areas

Three of the functional areas (intelligence, operations, and logistics) display a menu with the following choices:

-- View Map - Displays a map chosen by the user.

-- Plot Map Data - This module, when implemented, will display a user selected map and will allow the user to plot data on the map. The items to be plotted should be available from a large list selection of tactical, coordination, natural and man-made features. The data files will be updated by this function.

-- Extract Data - When implemented, this module allows the user to extract the location of selected features or to extract feature data of a given point. By entering grid coordinates, the user will receive all of the data listed for the location indicated. By specifying a search area and features to be looked for, (hill tops, bridges etc.), the user will receive the grid coordinates that meet the specifications.

-- Update Data - The implementation of this module will take coordinates and data specifications and alter the data records.

-- Exit - Exits functional area and returns to main menu.

The fourth functional area is slightly different, although they all may be different in the full implementation. The fire support area will perform the following functions when implemented:

-- List of Targets - This module will create or display target lists. Lists will be created by plotting on a map or by extracting from map files. The display mode will produce a text listing or a map display.

-- Coordination Lines - This module will display a map for review or updating of fire support coordination lines and information.

-- Maps - Situation - Displays the map produced by the operations function.

-- Maps - Intelligence - Displays the maps produced by the intelligence function.

-- Exit - Exits functional area and returns to main menu.

While each of the functional areas will utilize similar program code, each has particular data needs and interests. To separate the data of each, unique file names will be used for each use. For example, Do6008.dat and Mo6008.dat could be used for operations data while DL6008.DAT and ML6008.DAT could be used for logistics. Of more concern to the researchers than keeping data separate was the issue of data integrity. This subject is beyond the scope of this study but will have to be addressed in implementation.

Additional modules required should be determined by analysis of the users needs. The view of the researchers was that all desired major functions may be met, but all possible uses cannot be accommodated. The automated system cannot match the flexibility of the human with a paper map. It can though provide services not available, such as three dimensional views.

5. View Map

The view map module, actually the CREATE.EXE file, as implemented provides the user with a choice of displaying an old map, creating a new one or exiting back to the functional area menu. If the user chooses to display an old map, the four digit coordinate of the lower left hand corner is requested and the 'M' file is displayed. In the full implementation the maps available on the current or selected drive would be listed, from which the user may select. If the user is creating a new map the four digit coordinate is requested, the 'M' file is created and the map displayed. In the final implementation, the old map routine will have

to make allowances for outdated 'M' files, possibly through their destruction when any of their component 'D' files are altered. The old map usage is important in that the speed of the new map creation may take considerably longer.

The display issues were of primary concern in developing the prototype. These issues are more complex than those of mere data manipulation. Two questions in particular needed to be answered; how much data could be displayed and, more importantly, could the data be intelligently displayed? Enhanced Graphics Adapter (EGA) mode was utilized because of the increased resolution over Color Graphics Adapter (CGA) and the importance of color, not available in monochrome mode, as a tool to convey information.

The EGA display has 640 pixels horizontally and 350 vertically. Conceptually, this allows a maximum of 224,000 points or 560 square kilometers of 400 points each. To pass comprehendible information though, some system of symbols must be used. The standard text characters are eight pixels by fourteen, allowing for 2000 separate points on a screen of five square kilometers. The standard character set was used in the prototype. It was successful at passing the information for each point but had three limitations. First, the standard text characters obviously do not look like map symbols. Additionally, the eight by fourteen pixel arrangement distorts the square grid into a rectangle. Most importantly, five square kilometers is an unacceptably small section of map. To correct these limitations, square, user defined, map characters must be created and used. First, the determination must be made as to the size required for each point. The smaller the character is, the less capable we are of creating unique distinguishable symbols. A larger character means greater flexibility for designing symbols but less map coverage. A four by four or five by five pixel

character would allow for sufficient character set size without points becoming too course or too small. This would provide for thirty-five to twenty-two square kilometers to be represented at any time.

The possible four by eight kilometer map, achieved by using a four by four pixel character set, does not provide an adequate size map display. An alternative method to increase the point density would be to display the common information from more than one point as a single symbol. To do this, when a format file was being created, a point would have to be compared with its neighbors to determine if they possessed the same characteristic. This would increase the number of points that could be displayed but requires a significant increase of comparisons and memory accesses, slowing down processing. A combination of the smaller characters, group symbol representation and an improved, higher resolution, monitor may be necessary to produce a satisfactory display. If a solution cannot be reached, the value of an automated mapping system without graphic displays must be assessed.

Further complicating the display problem are people's perception of what features should look like on a map. A road, railroad, or electrical line is thought of as a continuous line. Using the same character for North - South roads as East - West roads fails to produce the continuous line representation desired. Road junctions should look like road junctions and bridges should be oriented the same direction as the roads they support. To make this happen will require three things; more than one character per feature must be available, memory must be held of the features around a point, and the logic must associate the neighboring features and pick the appropriate character. EGA is capable of using 1024 characters, but the memory and logic requirements will be difficult to meet adequately.

Without a finer display, the result still will not be smooth since characters cannot be made for orientation to every point of the compass.

The use of color proved to be extremely beneficial. Color was used to indicate water and changes in elevation. Color served this function well since few memory variables were needed to keep track of base points. Their numeric values were then used for incrementing and decrementing elevation strata. The alternative use of contour lines would have required the storing and comparing of each four neighboring points. This would be needed to determine when and where lines were to be drawn for each relationship of a point to its neighbor. This is not only more time and memory intensive but also confuses the map symbol issue as well. Two possible advantages of utilizing contour lines would be to free color to represent other uses or to show elevation on a monochrome display. The use of color for elevation contouring is in the researchers' opinion the best use of this asset with the test hardware.

D. EVALUATION

The prototype operated as desired and indicated that the development of a usable system is technically possible. The use of one square kilometer data files and the use of color coding of elevations were concluded to be sound approaches. The researchers also determined that the microcomputer hardware used was not capable of producing a display of a sufficiently large area. If a display capable of giving a large map coverage is used, it is conceivable that the extraction of the data for the display will be unacceptably slow. Smaller less environmentally demanding equipment than that used with the prototype, such as laptops, would be incapable of supporting the implemented system at this time. Accordingly, use of the envisioned system below the

battalion level is not practical and the use of the system will be limited at all levels by environmental conditions.

V. HARDWARE CONSIDERATIONS

The ability of microcomputers to do the necessary calculations, the ability of storage media to survive the environment and hold enough data to be useful, and the capabilities of current display technology to provide adequate resolution, are a few of the most important hardware questions that need to be answered if digital mapping systems are to be implemented for general use.

A. CPU CALCULATING SPEED

The researchers came to the realization during this study that the CPU of a computer and it's calculating capabilities are extremely important to a project such as this. The time required to generate the graphics screens, sort the data base, combine the data sectors, and do the matrix calculations to present map overlays will weigh very heavily on the computer and its CPU. It is suspected that a complete implementation of a TDMS would be practically unusable on a PC, an XT, or one of the portables that run an 8088 or 8086 at 4.77 MHz. The "turbo XT" clones would be very little better under most loads of this system. In fact the AT and "turbo AT" clones might prove sluggish in many cases. However, there is a ray of sunshine in the new 80386 CPU. It is relatively new technology, and as such the boundaries of its capabilities are just beginning to be explored.

1. PC's, XT's, and Clones

The IBM PC was announced 12 August 1981 and the PC XT on 8 March 1983. These machines started a revolution in the world of computing. It provided the ability to increase productivity of the user, while not requiring any more (and

often times much less) work. But, the engineers were conservative in their designs for many reasons. The components (specially memory chips) were extremely expensive, which made justification of a state-of-the-art design difficult. Therefore, a leisurely speed of 4.77 MHz was used on the eight bit bus for the Intel 8088.

2. IBM AT, AT Compatibles, and Turbo AT's

The IBM AT was announced 14 August 1984, which boosted the personal computer into another dimension; the Intel 80286 CPU. This vastly improved cousin of the 8088 which was the heart of the PC and XT, ran more efficiently through improved micro-circuitry and improved micro-code as well as the awesome speed of a 6 MHz clock chip. The AT's sixteen bit bus could handle more complex instructions more efficiently producing twice the throughput.

Hackers soon discovered that the 80286 would run faster if the 12 MHz clock chip (which is stepped down 50% in circuitry) was replaced with a 16, a 17, an 18 or even up to a 20 MHz chip. The clone makers quickly responded with technology similar to the turbo XT's, producing the turbo AT's some of which are now being marketed as 14 and 16 MHz models.

3. 386 Machines

The Intel 80386 CPU, a 32 bit member of the 8088 family was another order of magnitude faster in computations than even the 80286. The faster clock allowed a more efficient CPU to handle more 32 bit words and instructions in one second. "Comparing its Deskpro 386 with an AT, Compaq says it's two to three times as fast (the PC Magazine Labs benchmark test confirmed it's at least twice as fast as a PC AT) and as much as ten times as fast with 32-bit software."(Howard and Wong, 1986, p. 136)

But the speed has increased even more with the recent release of 20 MHz 386 machines, and there are rumors

of 25 MHz machines in test, not to mention the rumors of the 80486 processor nearing completion of design. The Compaq uses 128 K of RAM, "...to keep a copy of the EGA ROM BIOS that supports the EGA. Access to the EGA using the RAM-based BIOS is faster because the copy resides in 32-bit memory, which runs twice as fast as memory that is located on the 8 or 16-bit bus. Some video operations can be up to twice as fast."(Howard and Wong, 1986, p. 138) The recent announcement of an IBM 386 based machine may further spur development that will push technology forward.

Other recent advances in technology have improved the possibility of a TDMS's success. The release of Phar Lap Software's 386/ASM/LINK, which fully supports the 32-bit capabilities of the 386 machines is one example. This will allow any programs written in or assembled under this assembler to use the faster 32-bit memory data path for much faster (as much as ten times faster according to Compaq Corp.) operations. The set of development programs in this package will provide tools to improve and speed program development on the 386 architecture.(Trask, 1987, pp. 224-227)

Thus the state-of-the-art technology which offers the kind of speed that may be required to operate a TDMS in its full implementation at reasonable speeds is available. It would at the very least be a very dramatic improvement over the systems used to develop the prototype which were state of the art at the beginning of this study. We cannot imagine what tomorrow may bring, but surely it will improve the operating speeds for a TDMS.

B. STORAGE MEDIA

Storage media used in any project is dictated by the equipment used and the environment it must endure. There are many types of media that are capable of providing the

storage necessary for a digital mapping system computer in the field, but not many that could survive the environment. The media used would have to be able to endure every extreme in rough handling, humidity, dirt and dust, moisture, and temperature. These conditions would prove deadly to many means of data storage. However, several media available today would survive this kind of treatment and provide usable data for a TDMS.

1. Defense Mapping Agency Data Storage

Presently DMA data is available only on nine-track tape which is not a good media for field use, or even for microcomputer use. The TTADB data for a single grid measuring 1 kilometer by 1 kilometer (or 80 data points by 80 data points) requires 524,902 bytes of storage space. This data can be significantly trimmed by removing many unused or unnecessary data bits, at which time it could easily be transferred to any chosen storage media. But the data for one full-screen map display of 100 square kilometers in the format defined in Chapter II, will still require 240 KB storage space.

2. Floppy Disk

The TDMS data storage requirement for one kilometer square grid is 3400 bytes.

TABLE 3. GRID STORAGE CAPABILITIES

Media	Size	No. Grids
5 1/4 in HD Disk	1.2 MB	370
3 1/2 in Disk	720 KB	216
3 1/2 in Disk	1.44 MB	444
Hard Disk Drive	20 MB	6168
CD-ROM Disk	550 MB	169,622
WORM Drive	200 MB	61,680
Laser Card	2 MB	615

This means that one CD-ROM will hold map data for terrain the size of the state of Washington.

This would indicate that storage of a standard 1:50,000 map in digital form would not be optimized on a floppy disk of any above mentioned size. Yet with multiple disk sets the space is available on floppy disk to provide these digital maps.

> Floppy disks and tape share the characteristic of operating with the head in contact with the medium. Floppy disks were originally intended to load programs for mainframe computers and did not need to survive many passes. They are presently the main medium for microcomputer archival storage. This makes friction and wear characteristics very important. High-quality floppy disks are now burnished, in the manner of Winchester disks to assure smoothness and absence of particles that might break off and scratch the disk. Heads are made of hard ferrites embedded in hard ceramic sliders.
>
> The remaining culprits for floppy failures are dirt and physical damage. All these problems are solved by the newer 3 1/2-inch floppy format, which uses a rigid jacket, a sliding metal shutter to cover the access slots, and a metal hub attached to the plastic disk.(Laub, 1986, p. 168)

A floppy disk, of any ilk, will succumb to the heat, dirt and dust, or moisture of the field environment in a very short time.

3. Hard Disk (Winchester Drives)

The most familiar storage media that offers the needed storage space is the hard disk (or Winchester drive), which has proven since their use in microcomputers began, to be quite durable in an office environment. However, a hard disk as we know them today, would probably not survive more than a few days in the field because of critical power requirements and the delicate nature of the platters. "With tolerances slightly above the angstrom level, dropping a chassis a quarter-inch, or tapping it with your toe, is the hard disk equivalent of an atom bomb going off directly overhead."(Somerson, 1987, p. 200) One gentle bump is

enough to cause the Winchesters "flying head" to crash into the platter, permanently destroying the data at that position, and possibly destroying the entire drive unit. One power surge or voltage drop can have the same results.

A hard disk is the fastest media that has the capability to store that amount of data necessary for TDMS with some average access time at 13 mili-seconds. It also offers the ability to both read and write data which is an absolute necessity for the TDMS as described in Chapter IV.

4. Tape Cartridge

A tape cartridge might be acceptable. Having a thick aluminum base plate and a hard plastic case with a door to protect the tape from dirt and damage it might survive the harsh environment. Some tape cartridges store over 100 MBytes in paperback book size cases, while the smaller "DC2000 cartridges... let you tote 40 megabytes of formatted data in your shirt pocket."(Rosch, "DC2000 Systems: Pocket-Size Backup", 1987, p. 111) Yet the primary purpose of a tape cartridge unit is as a hard disk drive backup system. Not being designed for random access storage tape units are very slow finding specific data, but are quite good at sequential storage methods. In conjunction with microcomputers a tape unit normally has another medium (a hard drive) to transfer the tape data to, without which the tape is very clumsy or even impossible to use. A hard drive has already been ruled out for a field deployable computer because of its short life expectancy, so a tape unit may also be considered unacceptable.

5. Integrated Circuit Card

An IC Card is a credit card size unit that contains one or many integrated circuit(s) physically in the card. These however, are limited to 64 KB of information in a unit that is all memory. This is inadequate storage for any but a minuscule digitized map data base.

6. Optical Storage Media

The most promising technology is laser optical storage media. Optical storage media fall into four catagories; Erasable Optical Disks, CD-ROM (Compact Disk - Read Only Memory), WORM (Write Once - Read Many) disks, and Optical Memory Cards.

> The technology itself, which uses finely focused laser beams to cram at least 50 times more data onto a given number of square inches, may bring many mainframe-scale applications onto your desk.
>
> Apart from their data density, optical media have other inherent advantages. Because they are read by a light beam, the reading mechanism can be considerably farther away from the recording surface, making head crashes a thing of the past. The surface can also be protected from physical damage like fingerprints and scratches by a transparent plastic coating. And the incredible bits-to-burn data capacity means that data can be recorded with a great deal of redundant error-checking information, so that even if a part of the disk (media) is physically damaged, actual data loss can be minimal.(Helliwell, 1986, p. 149)

This would indicate that optical media can get dirty, be treated roughly, or get wet and still be usable if some minimum measures of cleanliness (such as wiping off dirt and moisture before inserting the disk in the drive) are observed. The tremendous amount of storage made available would store a vast amount of map data to produce a very large map display set.

a. Erasable Optical Disk

The erasable optical disk is still in its infancy. Much research and development is being carried out with some companies in design and testing stages. Many experts believe that in the 1990's this technology will be common place. The CD-ROM and WORM drives are readily available today from a number of manufacturers and integrators.

b. Compact Disk- Read Only Memory (CD-ROM) Disk

"A CD-ROM can hold about 550 megabytes of usable data--equivalent to over 400 of the high-density 1.2 megabyte disks used by the PC AT or to over 1,500 old-fashioned 360K-byte disks."(Helliwell, 1986, p. 150) In his description of optical disk storage of the United States Census Bureau's Dual Independent Map Encoding (DIME) digital map data system, Donald Cooke said:

> Including the number of characters needed to store street names, node numbers, and so on, each segment requires under 100 bytes of storage. The nation-wide file will therefore contain between 1 billion and 1.5 billion bytes of data. The 100-byte record size is generous; applying some compression tricks could reduce this by more than 50 percent. For example, the high end of an address range can be represented relative to the low end, saving a couple of bytes, or a central street-name inventory can serve all the street segments, rather than repeating the street name in each record. The net result is that it is possible to compress a digital street map containing all the streets in the country to fit comfortably on one 550-megabyte CD-ROM disk.(Cooke, 1987, p. 132)

These drives are quite inexpensive, and are currently available for under $1000 for a CD-ROM drive and under $2500 for a WORM. But the disks are another matter.

Custom CD ROM applications demand a hefty investment. Lewis Miller, vice president of marketing at Knowledge Access, an electronic publishing firm in Mountain View, California, estimates that transferring 100 MB of data to CD ROM from tape or floppies could cost nearly $20,000. "Copies of the master disk cost $25 each for 100 or more, or $10 each for 1000 and up."(Cummings, 1987, p. 232) So in large quantities the CD-ROM is an economical medium, but for small production runs such as a single 1:50,000 map data set for one operation the cost would be staggering.

One solution would be to develop the disks on a large scale and store them for distribution as needed, just as paper maps are today. But that would cause distribution problems. Prepositioning disks from a large production run might work, but there are problems inherent in that system

also. Other problems with CD-ROM are that it is "Read Only Memory" and cannot be written on by the users, and the access times are slow when compared to hard drives.

c. Write Once- Read Many (WORM) Drive

WORM (Write Once Read Many times) offers an acceptable solution. A WORM drive is capable of storing between 120 and 200 megabytes depending on the manufacturer. "The claimed life for WORM optical cartridges is 10 years, versus 3 years for magnetic media. As a result, you can expect the data on a WORM cartridge to endure longer than the expected life of the host computer."(Rosch, "WORMs for Mass Storage", 1987, pp. 135-136) A WORM disk presently costs $125 and up for single sided media, and up to $249 for double sided disks, which can be produced as required on any system with a WORM drive.[11]

Either the WORM or CD-ROM drives would offer adequate support of the very large databases required to support a digital mapping system. However, the WORM offers the capability to write to the disk which the CD-ROM does not. This would give the TDMS the ability to construct its composite maps files on disk, and would give the user units the ability to create overlay files as they required.

d. Optical Memory (Laser) Card

An even smaller more compact media alternative for smaller map data areas is the Optical Memory Card. An Optical Memory Card is:

> A card based upon a unique optical recording medium similar to that used in today's optical disks. But, unlike optical disks, the optical memory card is a convenient size and shape (similar to a typical credit card) and is very inexpensive.
>
> One such optical memory card, the Drexler LaserCard™ can store up to 16 million bits of updatable but nonerasable digital data at a price well under $5 per card.(Barnes and Sukernick, 1986, p. 5)

---

[11]. Figures derived from "WORMs: Summary of Features" Table in Rosch, "WORMs for Mass Storage", p. 142.

Several data sizes are available as need requires. There are primarily three different cards: a 625 KB card with one 16mm strip of recording media for small jobs; a 2.2 MB card with two 35mm strips of media on one side; and then there is the 8.8 - 10 MB card that has all the surface of both sides covered with media. Each of these cards would provide adequate storage for some particular map data sets that they would be used to transport.

The cards are produced at the Drexler plant (the sole source of the cards) using photolithography to very quickly reproduce literally thousands of identical optical memory cards. These cards can then be updated or blank cards can be written at a user site by reader/writer units which are attached to the host (micro or mini) computer by means of a SCSI (Small Computer Standard Interface) or RS-232c (serial) interface. "A moderate-speed laser recording system operating at 100 kilobits/second could produce one 2-megabyte LaserCard™ on-demand in less than three minutes... and these same optical memory cards could later have data added to them by low-speed laser recording (10 kilobits/second)."(Barnes and Sukernick, 1986, p. 5) These cards could be used, customized and updated as required at the local unit level, using a read/write unit internal to (the Nipponcoinco LC-301 reader/writer weighs 5.5 lbs and is about the size of a full-height disk drive)[12], or attached to the microcomputer that provides the map display. There is even an existing security system that can be implemented in each optical memory card that virtually assures that the data could not be used by unauthorized personnel.

C. MONITOR (DISPLAY)

We determined early in the study that color was a necessity to provide the information necessary on a map

---

[12]. "Optical-Card Reader Uses New Technology", PC Week Magazine, (March 31, 1987), p. 22.

display. While the Hercules Monochrome Graphics Adapter provides a slight advantage in resolution over the color systems, the trade-off was well worth it for the increase in the amount of information represented, and the ease of understanding it afforded. Therefore, the TDMS prototype system was developed using an Enhanced Graphics Adapter (EGA card) and an Enhanced Color Display which subscribe to the IBM Enhanced Graphics Adapter standard. This means that the card will display 16 colors at a maximum resolution of 640 X 350 pixels per screen.(Shneiderman, 1987)

We found that EGA is just barely adequate for the task at hand, and an order of magnitude improvement in monitor resolution would heighten the map display greatly. This would allow more accurate placement of the symbols on the screen to improve resolution of the map. It should also allow a larger variety of colors to be used for better distinguishing between symbols and conveying information. With a greatly improved monitor the capability variable display resolution could improve usability of the system.

1. Hercules Monochrome Graphics Adapter (Hercules Graphics)

"...the Hercules Graphics Card, a monochrome graphics adapter that displays graphics on a standard monochrome monitor at a resolution of 720 dots by 348 lines."(Alsop, 1986, p. 141) But this resolution is in two colors, background and foreground. As explained in the Schneiderman text, this will not present enough usable information quickly enough even at this resolution. Colors are necessary unless each symbol can be represented by a distinctly different and recognizable shape. To be effective in monochrome TDMS would need much improved resolution, possibly two or three orders of magnitude improvement.

2. Color Graphics Monitor Adapter (CGA)

The color graphics adapter proved to be unsatisfactory. In alphanumeric mode only characters may be displayed in one of 16 colors at a resolution of 160- by 100-pixels. The all-points-addressable (APA) modes provided improvement, however the limitations were too strict.

> In the APA graphics mode, there are two resolutions available; a medium-resolution color graphics mode (320 PELs by 200 rows) and a high-resolution black-and-white graphics mode (640 PELs by 200 rows). In medium-resolution mode, each picture element(PEL) may have one of four colors. The high-resolution mode is available only in black-and-white....(International Business Machines, 1984, p. 3)

None of these options are adequate to represent the information it is necessary to depict on a map screen display. Therefore, CGA was not ever considered as an alternative for the system display.

3. Enhanced Graphics Adapter (EGA)

The EGA system provides a vast improvement over CGA with 16 colors at a resolution of 640 by 350, almost matching the resolution provided by the Hercules Graphics Adapter, while providing the colors necessary.(Alsop, 1986, p. 142) This was adopted as the minimum acceptable display and adapter combination for prototype development because EGA was commonly available, and it provides the capabilities necessary to make a TDMS usable.

Of the 16 available colors 8 were used in the prototype of the map display, which are clearly differentiable as representing different elevations.

4. Professional Graphics System (PGS)

"The IBM Professional Graphics System (PGS) consists of a high-resolution color monitor matched to a graphics-controller board; it provides flickerless 640- by 480-pixel graphics images in 256-out of a possible 4096- simultaneous

colors."(Jadrnicek, 1985, p. 355) This is a higher performance display which is also more expensive than EGA.

5. Multiscan Monitors

The new multiscan monitors (ie. NEC Multisync, Sony Multiscan, etc.) have the capability of providing better resolution than even the PGC system. "The maximum resolution claimed is 900 by 560, compared to NEC Multisync's 800 by 560 and the EGA standard of 640 by 350."(Wiswell, Puglia, and Rose, 1987, p. 136) The multiscan monitors derive an improvement in resolution from an increased capability in the horizontal scan rate of the electron gun, and in increased bandwidth that will allow them to read the output adapters up to that speed. The best recognized of these new monitors is the NEC Multisync which has a horizontal scan rate of 35MHz, and can therefore synchronize with the 14 MHz of CGA, 16 MHz of EGA, and the 30.48MHz of the PGC adapters output signal.

However, it has been in only the last three or four months that adapter cards that can produce the display resolution to challenge the multiscan monitors have come onto the market (ie. the NEC GB1 and the Genoa SuperEGA adapter cards). These adapters were not available the researchers and therefore comment on their ability to display TDMS satisfactorily cannot be made. The NEC Multisync with the proper adapter card could produce seventy one kilometer grid squares using four by four pixel characters. This is still far short of what is felt to be the optimum map display.

6. Graphic Workstations

We acknowledge that the resolution and colors available to the minicomputer's graphic workstation are adequate to produce the results that are desired for a TDMS. In fact the computing power of the minicomputer is desirable if not a necessity, however, our thesis is that a computer

can produce adequate map display's to make them a usable tool to a tactical field commander. The minicomputer is not the answer because of the rough handling and limited environmental support that is characteristic of field operations.

7. Video Co-processors

The recent development of adapter cards and software to support the new dedicated "video co-processors" of Intel and Texas Instruments will further the cause of display resolution. In fact, the Intel 82786 graphic processor "... can display 1024 colors simultaneously and supports resolutions from 640 by 480 pixels by 8 bits, ...up to a maximum of 4096 by 4096 pixels by 1 bit."(Nichols, 1987, p. 135) The adapter cards built around this chip will provide much faster screen draw and re-draw, better resolution, and more colors to enhance a TDMS. Unfortunately, this technology is not readily available today. But it will be by the time a fully developed TDMS could be fielded.

D. CONCLUSION

The problem of inadequate data storage space on a microcomputer system for the TDMS map database can be solved with the optical storage technology available today. A single CD-ROM, WORM disk, or Laser Card provide the advantages of tremendous storage space (over 616 one kilometer grids on a laser card), small and convenient size, virtual indestructibility, and current availability. This infant industry will build upon these every day as this technology matures.

The improvement in display resolution that is required to make TDMS a true productivity tool may be derived through three means: (1) larger screens, (2) smaller pixels which allows more on the screen, and (3) higher scan rates. This technology is progressing as is indicated by the advances

over the last three years from CGA, to EGA, to PGA, and now into the multiscan monitors and their graphic adapters.

Yet according to Rich Bader, co-founder of Intel Corporation's Personal Computer Enhancement Operation; "What's happening is that you want more pixels on the screen in order to get better resolution. Clearly, the more pixels you add, the more CPU cycles it ordinarily takes to keep the screen refreshed."(Knorr, 1987, p. 222) This indicates that more machine is required to support improved graphics. The present microcomputer technology does furnish the capability to improve over the next few years, because of the relatively new 80386 CPU, new graphics co-processors, 32-bit compilers and assemblers.

VI. EVALUATION

The standard tactical military map is a two dimensional representation of real terrain. The 1:50,000 map, most common in planning an exercise or mission, measures 40 inches by 52 inches, and weighs four ounces. This allows a standard representation of approximately 2700 square kilometers. It displays dozens of features and symbols in nine colors on a high grade sheet of paper with a resolution of 12,000 dots per inch. Without updating and reprinting of the map by DMA its representation cannot be changed. Any system for mapping must be compared to this standard.

A. DATA

In the data area of study, there were the following conclusions.

1. Feasibility

The TTADB is adequate and will be available from DMA for use in a digitized mapping system of world-wide scope. The lack of text data puts the digitized database at a disadvantage to the paper map. Unless a method for providing the names of rivers and towns, etc. in the database is developed, it cannot fully replace paper.

2. Limitations

The TTADB stores data in 199 bit records. There are 6561 records per square kilometer. This means a 1:50,000 map contains approximately 438 megabytes of information. This amount has been greatly reduced in our prototype system without sacrificing accuracy to an unacceptable point, but the database will still be extremely large.

Any system designed will be vulnerable to its data source. The display is capable of creating all of the

symbols required by TTADB and more, but only that information contained in TTADB can be displayed. Any failure of the TTADB or the DMA in their production of the TTADB will result in a failure of the system. A microcomputer mapping system can only compete with paper maps if TTADB contains the same useful data, or more, as its paper counterpart.

B. SOFTWARE

In reviewing the current, related DOD initiatives and with the development of the prototype, there were the following conclusions.

1. Feasibility

Both the prototype and the MicroDEM programs indicated a digitized mapping system could produce traditional two dimensional map displays, but also map formats outside the range of paper maps, e.g. three dimensional and tinted slope maps.

2. Limitations

Scanning over large sections of a map and large search/sorts may be unacceptably slow. The production of field deployable systems will require a higher performance machine or more sophisticated algorithms.

C. HARDWARE

Conclusions as to hardware are limited by the hardware available to the researchers. Performance of better equipment is offered as mitigation only.

1. Feasibility

Both the prototype and the MicroDem programs indicated that the basic requirements for a digitized mapping system could be met with commonly available hardware.

2. Limitations

The micro computer used to develop the digital mapping system prototype (an IBM AT compatible) was 21 inches wide, 16 1/2 inches deep, and 6 1/2 inches high. It weighed 60 lbs including the keyboard and the monitor which was 15 inches wide, 14 1/2 inches deep, and 11 1/2 inches high. This may be unacceptable for field use since its superiority to a four ounce paper map is as yet unproven. Currently vendors are creating systems which are smaller and may be equally capable and more environmentally acceptable.

An EGA system is capable of representing 16 colors at one time, in a picture 7 inches by 10 inches. This allowed a representation screen of only five square kilometers in the prototype. While a maximum of 560 square kilometers are possible, the practical limit appears to be less than one hundred with this equipment. Monitors are currently available for microcomputers with greater screen densities, more colors and larger screens.

As previously noted, data storage and processor speed were considered to be limiting factors to system performance. It was felt that the 80286 processor and the disk technology used by the researchers may be only marginally adequate for the demands of an operational system. The 80386 processor and developing storage technology may eliminate this as an issue.

D. CONCLUSION

The utilization of digitized maps by ground tactical units is technologically feasible. The microcomputers used were considered to be marginally adequate for the task, given their limitations of processor speed, storage, and display capability. Currently available advances in microcomputer hardware should reduce these limitations. Accordingly, the implementation of a microcomputer mapping

system is considered to be viable with current technology. It is recognized that the computer cannot replace the paper map. However, it can be a valuable tool for both analysis and the high speed extraction of data.

APPENDIX A

TERRANAL: MICROCOMPUTER TERRAIN MAPPING PACKAGE

(The TERRANAL description is extracted from a paper by Major Peter L. Guth (USAR) submitted for March 1986 meeting, American Congress on Surveying and Mapping.)

The TERRANAL program requires the following: an MS-DOS computer with at least 256K of memory, one double sided disk drive, and a graphics card and monitor. Optimal execution requires either a hard disk, or an additional 180K of memory to place a complete data set in memory. Dot matrix printers provide hard copy output using the DOS screen dump facility, and IBM or Epson printers support greater resolution on certain functions.

TERRANAL is user-friendly, menu-driven, and error-trapped, with optional help screens. TERRANAL accepts two coordinate systems for describing location: latitude and longitude, and UTM. The UTM grid is present, if inconspicuous, on most USGS maps, and easier to use than minutes and seconds for latitude and longitude. Although UTM coordinates represent the primary reference for program users, we retain our data base in latitude-longitude for the following reasons: (1) we want to remain as close to the standard data DMA base as possible; (2) we can avoid the problems of grid zone boundaries in the data set and not provide duplicate data sets along the boundaries; and (3) we prefer the theoretical basis of interpolating between the DMA values rather than interpolation between a UTM data set that has already been interpolated. Thus we perform the UTM to latitude-longitude conversion within the TERRANAL program, but only for the end points, and then use a linear interpolation within the data array. This assumes a flat earth over that portion of the data set in use, which introduces insignificant error. From one end of a 15 minute

map sheet to another, data spacing varies by a half meter, which is negligible considering the accuracy and resolution of the data.

The TERRANAL program currently performs the following functions:

-- Oblique 3D views. The computer will draw an oblique block view, looking in any desired direction.

-- Line of sight profiles. The computer will draw a topographic profile between any two points, and determine inter-visibility.

-- Cross section profiles. On IBM or Epson dot matrix printers, the program can draw topographic profiles at any desired scale and vertical exaggeration.

-- Contour maps. The computer will plot contour maps. For military purposes, the program can superimpose masked and visible locations for an observer at an arbitrary point.

-- Perspective 3D views. The second type of three dimensional view shows a camera-like perspective of a wedge-shaped piece of terrain.

-- Tinted elevation maps. Using seven elevation categories which the user can vary, the computer will draw tinted elevation maps. The size of regions allows color patterns to give seven categories with only four colors.

-- Slope maps. The user can select the method of slope calculation and boundaries for the four categories, and get a map of slopes within the area. Rapid slope changes allows only solid color categories.

-- Elevation histograms. The computer can rapidly compute the distribution of elevations within the data set, and statistics of the distribution.

-- We are developing additional capabilities. Because of the modular, structured nature of the Pascal source code, a user could easily incorporate desired improvements.

-- Vertical exaggeration can be selected for the line of sight, oblique, and perspective views. Long lines require vertical exaggeration to preclude flat profiles. The computer will automatically suggest the largest vertical exaggeration that will fit on the screen.

-- The oblique and perspective views are drawn as a series of profiles, progressing from rear to front. An area fill routine erases the hidden lines, precluding the calculation of hidden line.

TERRANAL serves as a tool for the analyst, who must clearly understand the limitations of the system. The data resolution (60-90 meters) means that the data accurately

reflects the general features of the topography, but will not depict micro-relief. The DTED accuracy must also be considered; our experience indicates that the accuracy faithfully portrays the general trends of the topography. Finally, vegetation and other features data are generally not available in digital format. Even with its data limitations, TERRANAL offers powerful digital terrain mapping capabilities on today's most widely available microcomputers.

For more information contact:

Capt. Gene Ressler USArmy
Dept. Geography & Computer Science
United States Military Academy
West Point, NY 10996-1695
Comm: (914) 938-4866/3128
AVON: 688-4866/3128
DDN/ARPANet: Ressler@WestPoint.ARPA

APPENDIX B

MICRODEM: MICROCOMPUTER PROGRAM FOR MANIPULATING LARGE DIGITAL TERRAIN MODELS

(The MicroDEM narrative was edited from documentation included with the program by Peter L. Guth, Eugene K. Ressler, and Todd S. Bacastow.)

Two serious impediments have limited the widespread use of digital cartography. First, the effort required to digitize topography limits application of well known principles. Second, existing programs require a high degree of computer sophistication and access to relatively large computers. We have developed a program, MicroDEM (for Microcomputer Digital Elevation Modelling), which addresses both impediments. MicroDEM runs on MS-DOS microcomputers (IBM PC and compatibles) and manipulates digital elevation models available from the U.S. Geological Survey. Digital data covers the entire United States; similar data might be available for other regions of the world. Thus we have an easy to use program running on the most popular and widely used microcomputer, using readily available digital data.

MicroDEM requires an MS-DOS computer with a graphics monitor conforming to the IBM Color Graphics Adapter (CGA) standard. (It will run on, but not yet use the greater capabilities of the newer Enhanced Graphics Adapter (EGA), and will run on some monochrome graphics monitors). Hard copy output can be generated on any device that implements the MS-DOS graphics screen dump utility. The program makes special use of Epson and IBM dot matrix printers to produce high resolution images.

The source code for MicroDEM (currently about 6500 lines) is in Turbo Pascal (Borland International, 1985). We chose this language because it has become a de facto standard: the compiler is inexpensive, it provides an

excellent development environment, it compiles rapidly and produces good executable code. Further, Pascal is transportable, and as a structured language it provides for easy program maintenance. As proof of these advantages, we have a version of the program running on an enhanced Apple II+. Because of the modularity, the program can be adapted to use digital data for a wide variety of functions.

MicroDEM uses the Digital Terrain Elevation Data (DTED) Level I produced by the Defense Mapping Agency. MicroDEM can be modified to work with other digital data bases, but none currently provide as widespread coverage. MicroDEM uses only digital elevation data; we have not yet expanded it to use digital feature data.

The MicroDEM program is user-friendly and error-trapped, so that the user knows what the computer wants and incorrect responses are handled predictably. The main menu offers the choice of the options available. In addition, the user can change the environment defaults, store and restore images produced by the program, or switch to a new area.

The MicroDEM program will currently produce the following:

-- Color tinted elevation maps, with seven user defined categories, at any desired scale;

-- Slope maps, with four user defined categories, at any desired scale;

-- Contour maps, at any desired scale and contour interval, and masked area plots showing terrain hidden from an observer;

-- Topographic profiles between any two desired points, at any desired scale and vertical exaggeration;

-- Oblique three dimensional views with any desired orientation;

-- Datum shifts and coordinate transformations between UTM and geographic.

MicroDEM provides a flexible means to manipulate digital elevation data available covering the entire United States.

The program runs on low-cost, industry standard MS-DOS microcomputers, putting digital cartography within the reach of almost any potential user.

Contact:

Dr. Peter Guth (Major USAR)
Dept. GeoScience
University of Nevada
Las Vegas, NV 89154
Comm: (702) 739-3262

APPENDIX C

AMPHIBIOUS OBJECTIVE AREA LAND MANAGEMENT SYSTEM

(The AOALMS Site Data Users Guide, documentation prepared for Naval Civil Engineering Laboratory under contract no. N00123-84-D-0152, was used to derive this explanation.)

1. System Description

The mission of the AOALMS is to enable U.S. Marine Corps (USMC) personnel to rapidly plan horizontal construction projects in the Amphibious Objective Area (AOA). The system consists of a microcomputer and software programs that aid USMC users in (1) rapidly examining site conditions and identifying promising locations for AOA facilities, (2) designing the facilities and calculating earthwork requirements, (3) determining the numbers and types of construction equipment needed to achieve facility completion dates, and (4) scheduling the entire AOA horizontal construction effort to ensure optimum usage of construction equipment. The AOALMS is used by the Landing Force Commander and USMC Engineer Support Battalion for the planning and construction of facilities worldwide. The planning is accomplished prior to departure (mount-out), aboard ship, and after arrival in the AOA.

The AOALMS is currently designed for operation on IBM XT and AT microcomputers (essentially, IBM Personal Computers (PC's) with hard disk drives). The IBM XT consists of a monitor, one 5-1/4-inch floppy disk drive, one hard disk drive, and a keyboard. The IBM XT uses an Epson FX-100 dot-matrix printer to print reports and to dump screen graphics. One program in the AOALMS requires the use of a large Gigatek monitor to provide high-resolution graphics.

Most AOALMS software has been programmed in Janus Ada, a microcomputer version of the Ada programming language. The program that uses the Gigatek monitor is coded in Basic. The AOALMS has been developed using the PC-DOS operating

system, the standard operating system for the IBM PC family of microcomputers.

The programs consist of two independent subsystems: the Site Selection Subsystem and the Site Evaluation Subsystem. Within the Site Evaluation Subsystem are four modules: (1) Site Data Module, (2) Facility Module, (3) Equipment Module, and (4) Schedule Module.

The subsystems and modules perform specific operations within the AOALMS. These operations are summarized in table below.

Subsystem and Module Operations

Subsystem/Module	Operation
Site Select. Subsystem	Displays site conditions for areas up to 100 square kilometers, permitting user to select promising sites for AOA facilities. Uses custom monitor to provide clear graphics. It is the only AOALMS program coded in Basic.
Site Eval. Subsystem	Evaluates required earthwork, equipment, and time necessary to complete the facility.
Site Data Module	Provides information on site conditions to be used in calculations in other modules.
Facility Module	Designs facility and calculates earthwork requirements. In its final configuration, the module will provide calculations for standardized facilities. Current version handles only one type of facility, the Vertical/Short Takeoff and Landing (VSTOL) Airbase.
Equipment Module	Uses standard earthmoving equations to calculate productivity of USMC and Naval Construction Force equipment under the specific conditions at the site.
Scheduling Module	Allows user to schedule theearthwork construction effort for all facilities at the AOA by correlating the results of calculations performed by the Facility and Equipment Modules.

2. System Operation

a. Site Selection Subsystem

The Site Selection Subsystem is a stand-alone program that displays site conditions for areas up to 100 square kilometers, permitting you to select promising sites for AOA facilities. The program uses two monitors. The smaller monitor is used to enter data to the computer in response to displayed questions. The larger monitor presents a perspective display of a 100-square-kilometer AOA. The approximate run time for the program is 30 minutes.

You are asked to select the facility type to be evaluated. The only facility handled by the current program is the VSTOL Airbase. All other user choices will be rejected. You are asked for the distance from the facility to the nearest quarry with subgrade material. You are also requested to select the type of military unit performing the construction project (e.g., Naval Mobile Construction Battalion). The program then asks you to select the desired view of the AOA. The program provides recommended view selections to help you make your decisions. You select the desired direction of view, the altitude, the distance from the AOA, and topography exaggeration, if any. If you desire, you can position a VSTOL Airbase on the AOA by specifying a rotation angle and the location coordinates for the facility.

A fullcolor perspective display then appears on the larger monitor with the specified view of the AOA and the VSTOL Airbase. The view consists of several grids, each of which represents a 200- by 200-meter area. The grids are colored to signify the relative effort required to build the facility at that location.

b. Site Evaluation Subsystem

The Site Evaluation Subsystem evaluates the earthwork, equipment, and time required to complete the facility. This program is used after a preliminary assessment of various sites has been performed with the Site Selection Subsystem. The approximate run time for the Site Evaluation Subsystem is 3 hours.

A menu provides you with a list of the four modules. You can select any one. The Facility Module designs the facility and calculates earthwork requirements. You are asked for the type of facility. In its final configuration, the module will provide calculations for 11 standardized facilities. The current version handles only one type of facility, the VSTOL Airbase. You are asked to provide the facility location (easting and northing coordinates) and the clockwise rotation angle from north.

The computer performs extensive facility calculations that take more than 1 hour to complete. When the run is completed, you are provided with a facility menu that allows you to examine various results of the facility calculations. You can list earthwork quantities, graphically view profile cuts, and/or display an aerial view of the facility.

The Equipment Module uses standard earthmoving equations to calculate the productivity of USMC and Naval Construction Force equipment under the specific conditions at the site. This module has a run time of about 1 minute. You cannot display the output from this module.

The Scheduling Module allows you to schedule the earthwork construction effort for all facilities at the AOA by correlating the results of calculations performed by the Facility and Equipment Modules. From the scheduling menu, select Case I. You are asked to select equipment by function or type and to provide quantities of each. After

you have supplied this information, the computer will perform all scheduling calculations in less than 5 minutes. A menu is displayed enabling you to view the scheduling results, including a tabular schedule, a Gantt chart, and so on. Return to the scheduling menu and select Case II. You are asked to select equipment by function or type and to specify facility completion times. As with Case I, a menu is then displayed enabling you to view the scheduling results.

The Site Evaluation Subsystem has other features. One of them is the Site Data Module. If you call up the module from the main menu, you can construct a new site data disk. The Site Evaluation Subsystem requires one site data disk (360K) for each 1,000 by 1,000 -meter area. One to four disks will be needed to perform calculations for a VSTOL Airbase at any given location. The Site Selection Subsystem requires one site data disk for a 10,000 by 10,000 -meter area. For the purposes of the AOALMS, this is considered the standard size for an AOA.

Additional details are available through:

Dr. Carter Ward
Code L64
Naval Civil Engineering Laboratory
Port Hueneme, CA 93043
Comm: (805) 982-5021
AVON: 360-5021

APPENDIX D

TTADB FORMAT

(The AOALMS Site Data Users Guide, documentation prepared for Naval Civil Engineering Laboratory under contract no. N00123-84-D-0152, was the source of this TTADB data format explanation.)

The first block of information is a 9 item header where each item is binary integer and right justified in 32 bit fields. Each item contains the following:

Item 1 = Southwest Corner Easting (in meters)
Item 2 = Southwest Corner Northing (in meters)
Item 3 = No. of Columns
Item 4 = No. of Rows
Item 5 = Interval between columns (in centimeters)
Item 6 = Interval between rows (in centimeters)
Item 7 = No. of Physical blocks per column
Item 8 = Grid Zone
Item 9 = Spheroid Code (1-8) designated as follows:

Clark 1880 = 1
International = 2
Clarke 1866 = 3
Bessel = 4
Evevest = 5
Walbeck = 6
Fischer = 7
WGS 72 = 8

The remaining blocks contain the elevations and corresponding features. The first 32 bits of each logical block contains the block count. The next 16 bits contain the relative count from the Southwest corner easting. Therefore, each elevation t#m in a column will have the same easting count. The next 16 bits contain the starting relative count from the Southwest corner northing. Successive elevation items in a block have an implied relative northing value which is 1 more than the previous value. For example, if the starting relative northing count for the block is 0, then the third elevation item has a relative northing count of 2. Assuming a row interval of 12.5 meters, then the third elevation item is 25 meters north of the SW corner northing.

The remaining bits in a block contain elevation items. Each elevation item consists of 199 bits where the first 16 bits contain the elevation and the other 183 bits contain the associated codes for the features. Each elevation item is fully detailed in an attached format description. Each logical block is terminated by at least 36 one bits.

PROTOTYPE FORT LEWIS-YAKIMA TRAINING AREA*
TACTICAL TERRAIN ANALYSIS DATA BASE (TTADB) FORMAT
FOR 1:50,000 SCALE PRODUCTS

Data Element	Bit Designation	# of Bits	Code	Value Represented
Surface Configuration Overlay				
Elevation (m)	1 - 16	(16)		
Slope (%)	17 - 20	(4)	0	No Data
			1	0 - 3
			2	3 - 10
			3	10 - 20
			4	20 - 30
			5	30 - 45
			6	>45
			7	Naturally and/or culturally dissected land (0->45) (Numerous small hillocks sand dunes, glacial debris, landfills, dumps, etc.)
			8-14	
			15	Open Water
Vegetation Overlay				
Type	21 - 26	(6)	0	No Data
			1	Agriculture (dry crops)
			2	Agriculture (wetland rice)
			3	Agriculture (terraced crop: both wet and dry)
			4	Agriculture (shifting cult
			5	Brushland/Scrub (<5m.high, nearly open to medium sp)
			6	Brushland/Scrub (<5m high, medium to dense spacing)
			7	Coniferous/Evergreen Fores
			8	Deciduous Forest
			9	Mixed Forest
			10	Orchard/Plantation (rubber, palm, fruit, etc.)
			11	Grassland, Meadows, Pasture
			12	Grassland with Scattered Trees Some Scrub Growth
			13	Forest Clearings (cutover areas, burns, etc.)
			14	Swamp(mangrove,cypress,etc)
			15	Marsh/Bog(peat, muskeg, etc)
			16	Wetlands (L.S.I., low-lying wet areas)
			17	Vineyard/Hop-garden
			18	Bamboo
			19	Bare Ground
			20-22	
			23	Open Water
			24	Built-up Areas
			25-29	
			30-63	Not Used

NOTE: On account of programming time limitations, this Fort Lewis-Yakima prototype is a condensation of the full proposed Tactical Terrain Analysis Data Base. Numerous data fields have had to be compressed, omitted, or specifically tailored to the Fort Lewis-Yakima terrain conditions to meet these limitations.

Data Element	Bit Designation	# of Bits	Code	Value Represented
Canopy Closure(%)	27 - 29	(3)	0	No Data
			1	0 - 25
			2	25 - 50
			3	50 - 75
			4	75 - 100
			5-7	
Tree Height (m)	30 - 33	(4)	0	No Data
			1	0 - 2
			2	2 - 5
			3	5 - 10
			4	10 - 15
			5	15 - 20
			6	20 - 25
			7	25 - 30
			8	30 - 35
			9	>35
			10-15	Not Used
Stem Diameter(m)	34 - 37	(4)	0	No Data
			1	0
(Note: CCM formula uses meters and these ranges were selected because they best correspond to the push-over limits of the vehicles for which we compute CCM)			2	.00 - .02
			3	.02 - .04
			4	.04 - .06
			5	.06 - .08
			6	.08 - .10
			7	.10 - .15
			8	.15 - .20
			9	.20 - .25
			10	.25 - .50
			11	.50 - 1.00
			12	1 - 3
			13	3 - 5
			14	5 - 10
			15	> 10
Stem Spacing (m)	38 - 41	(4)	0	No Data
			1-9	0 - 4.0 (by 0.5)
			10	4 - 5
			11	5 - 6
			12	6 - 8
			13	8 - 10
			14	10 - 15
			15	>15
Vegetation Roughness Factor	42 - 46	(5)	0	0
			1	.1
			2	.2
			3	.3
			4	.4
			5	.5
			6	.6
			7	.7
			8	.8
			9	.9
			10-31	Not Used
Undergrowth	47 - 48	(2)	0	No Data
			1	None
			2	Sparse
			3	Dense

Data Element	Bit # of Designation	Bits	Code	Value Represented
Tree Crown Diameter (meters)	49 - 51	(3)	0	No Data
			1-7	Not Used
(Note: Need if we intend to do inter-visibility or line-of-sight products; otherwise will remain zeroed out)				
Height of Lowest Branch (meters)	52 - 54	(3)	0	No Data
			1-7	Not Used
(Note: Same as above)				
Surface Material Overlay Type	55 - 60	(6)	0	No Data
			1	GW - Gravel, well graded
			2	GP - Gravel, poorly graded
			3	GM - Gravel, silty, clayey
			4	GC - Gravel, clayey
			5	SW - Sand, well graded
			6	SP - Sand, poorly graded
			7	SM - Sand, silty
			8	SC - Sand, clayey
			9	ML - Silt
			10	CL - Clays
			11	OL - Organic silts
			12	MH - Inorganic silts
			13	CH - Fat clays
			14	OH - Fat organic clays
			15	PT - Organic peat
			16	Snowfield/Glacier
			17	Rock outcrops
			18	Evaporite
			19-61	
			62	Open water
			63	Not evaluated (built-up a etc)
Qualifier	61 - 65	(5)	0	No Data
			1	None
			2	Boulder field
			3	Quarry, mine, diggings
			4	Bare rock, smooth
			5	Lava flow
			6	Dunes
			7	Loose
			8	Karst
			9	Lathyritic
			10	Permafrost
			11	Frequent stone or rock outcrops
			12	Dissected
			13	Metal/ore slag dump
			14	Tailings, waste pile
			15	Strip mine
			16	Rugged bedrock
			17-31	
State of the Ground	66 - 69	(4)	0	No Data
			1	Dry
			2	Approximately 50% saturation
			3	Wet (saturated)
			4-14	0 - 1.0 (by 0.1--fraction of soil moisture in top half meter at time of measurement or CCM
synthesization)				

Data Element	Bit Designation	# of Bits	Code	Value Represented
Depth of Surface Material (meters)	70 - 71	(2)	0	No Data
			1	0 - 0.5
			2	> 0.5
Surface Roughness Factor - Medium and Heavy Tanks (M-1 Abrams, M60, and T-72 Tanks, etc)	72 - 75	(4)	0	No Data
			1	0
			2	0.05
			3	0.1
			4	0.2
			5	0.3
			6	0.4
			7	0.5
			8	0.6
			9	0.7
(Note: One SR table will be needed for each vehicle type for which a CCM map is to be prepared)			10	0.75
			11	0.80
			12	0.85
			13	0.90
			14	0.95
			15	1.00
Surface Roughness Factor - Large Wheeled Vehicles (M35 Truck, etc)	76 - 79	(4)	0	No Data
			1-15	Not Used
Surface Roughness Factor Small Wheeled Vehicles (M151 Jeep, etc)	80 - 83	(4)	0	No Data
			1-15	Not Used
Surface Roughness Factor - Light Tracked Vehicles (M113 APC, etc)	84 - 87	(4)	0	No Data
			1-15	Not Used
Surface Roughness Factor Foot Troops	88 - 91	(4)	0	No Data
			1-15	Not Used
Depth to Bedrock (meters)	92 - 95	(4)	0	No Data
			1-15	Not Used

(Note: Need if we intent to support penetration or engineering studies; otherwise will remain zeroed out)

IV. Surface Drainage Overlay

Type	96 - 98	(3)	0	No Data
			1	Stream Channel (Dry Wash or Intermittent, e.g., Arroyo)
			2	Lakes, Ponds, Reservoirs
			3	Stream Channel (Perennial)
			4	Stream Channel (Subject to Tidal fluctuations)
			5	Trenched Stream/Irrigation Canal/Canal/Drainage Ditch
			6	Off-Route Ford (Entrance and Exit points connected)
			7	Dam/Lock
Gap Width (Bank to Bank (m))	99 - 101	(3)	0	No Data
			1	< 4.5
			2	4.5 - 18
			3	18 - 50
			4	50 - 100
			5	100 - 142
			6	> 142

Data Element	Bit # of Designation	Bits	Code	Value Represented
Bottom Material	102 - 104 (3)		0	No Data
			1	Clay and Silt
			2	Silty Sand
			3	Sand and Gravel
			4	Gravel and Cobble
			5	Rocks and Boulders
			6	Bedrock
			7	Paved
Height, right bank (m)	105 - 107 (3)		0	No Data
			1	< 0.5
			2	0.5 - 1.0
			3	1.0 - 5.0
			4	> 5.0
			5-7	
Height, left bank (m)	108 - 110 (3)		0	No Data
			1	< 0.5
			2	0.5 - 1.0
			3	1.0 - 5.0
			4	> 5.0
			5-7	
Slope, right bank (%)	111 - 113 (3)		0	No Data
			1	< 30
			2	30 - 45
			3	45 - 60
			4	> 60
			5-7	
Slope, left bank (%)	114 - 116 (3)		0	No Data
			1	< 30
			2	30 - 45
			3	45 - 60
			4	> 60
			5-7	
Water velocity, average (meters/ seconds)	117 - 118 (2)		0	No Data
			1	<= 2.5
			2	> 2.5
			3	Not Used
Water depth, average (m)	119 - 121 (3)		0	No Data
			1	< 0.8
			2	0.8 - 1.6
			3	1.6 - 2.4
			4	> 2.4
			5-7	
Dense Vegetation Along Stream Banks	122 (1)		0	No Data
			1	> 50% Segment Length

(Note: Normally closely spaced row of trees, which could possibly hinder stream crossing operations)

Data Element	Bit Designation	# of Bits	Code	Value Represented
Transportation Overlay				
Type	123 - 126	(4)	0	No Data
			1	Bridge - Road
			2	Bridge - Railroad
			3	Tunnel - Road
			4	Tunnel - Railroad
			5	Dual Lane/Divided Highway/ Expressway
			6	Highway/Road
			7	Railroad
			8	Airfield
			9	Inland Waterway
			10	Lock
			11-15	
Condition	127 - 129	(3)	0	No Data
			1	Good
Operational (Note: Need part of this field now, will need the rest if we intend to support air operations or on-route trafficability studies)			2	Fair
			3	Poor/Deteriorated
			4	Damaged
			5	Destroyed
			6	Abandoned/Dismantled
			7	Under construction
Qualifier	130 - 132	(3)	0	No Data
			1	Road Constriction> 4 meters
(Note: Now do all except the grade in excess of 3% for railroads)			2	Grade in excess: 7%for road or 3% for railroads
			3	Sharp curve, radius<30 meter
			4	Ferry Site
			5	On Route Ford Site
			6	Electrified Line
			7	
Length(meters)	133 - 138	(6)	0	No Data
			1	Unknown
(Note: For this Fort Lewis-Yakima prototype, length only refers to Bridges, Tunnels, Airfields, and other transportation types less than 100 meters long. All lengths greater than 100 meters are determined by the number of points used to digitize the length of the feature - 0.25 mm (approx. 0.01 inches) equals 12.5 meters on the ground at 1:50,000 scale).			2	0 - 10
			3	10 - 20
			4	20 - 30
			5	30 - 40
			6	40 - 60
			7	60 - 80
			8	80 - 100
			9	> 100
			10-31	Not Used
Average Width (meters)	139 - 142	(4)	0	No Data
			1	Unknown
(Note: For this Fort Lewis-Yakima prototype, width refers to any transportation type, except Railroads,less than 50 meters wide. All widths greater than 50 meters are determined the same as lengths greater than 100 meters			2	0 - 3
			3	3 - 4
			4	4 - 5
			5	5 - 7
			6	7 - 10
			7	10 - 20
			8	20 - 50
			9	> 50
			10	Not Used

- see above).

Data Element	Bit Designation	# of Bits	Code	Value Represented
Surface	143 - 146	(4)	0	No Data
			1	Paved
(Note: Need part of this field now, will need the rest if we intend to support on-route trafficability studies)			2	Hard
			3	Loose/Gravel
			4	Unpaved
			5	Natural Earth
			6	Grass
			7	Macadam/Asphalt/Bituminous
			8	Concrete
			9	Stone/masonry/brick
			10-15	Not Used
Highways and/or Roads:				
Type	147 - 149	(3)	0	No Data
			1	All Weather
			2	Fair/Dry Weather
			3	Cart Tracks
			4	Trails
			5-7	
Railroads:				
Type	150 - 152	(3)	0	No Data
			1	Normal Gauge, single track
			2	Normal Gauge, dual track
			3	Normal Gauge, multiple (2 or more) tracks
			4	Narrow Gauge, single track
			5	Narrow Gauge, multi-track
			6	Broad Gauge, single track
			7	Broad Gauge, multi-track
Passing tracks, sidings & yards (meters)	153 - 154	(4)	0	No Data
			1	Passing track >=280
			2	Siding >=280
			3	Yard >=280
Tunnels:				
Height (meters)	155 - 157	(3)	0	No Data
			1	3 - 6
			2	6 - 8
			3	8 - 12
			4	> 12
			5-7	
Bridges:				
Type	158 - 161	(4)	0	No Data
			1	Truss
(Note: Need if we intend to support engineer or on-route trafficability studies; otherwise will remain zeroed out)			2	Girder
			3	Beam
			4	Slab
			5	Arch
			6	Suspension
			7	Floating
			8	Cable stayed
			9	Cantilever
			10-15	Not Used
Movement	162 - 164	(3)	0	No Data
			1	Fixed
(Note: Same as above)			2-7	Not Used

Data Element	Bit Designation	# of Bits	Code	Value Represented
Overhead Clearance	165 - 166	(2)	0	No Data
			1	Unknown
			2	Unlimited clearance
			3	Possible obstruction to military traffic
Horizontal Clearance	167 - 168	(2)	0	No Data
			1	Unknown
			2	Unlimited Clearance
			3	Possible obstruction to military traffic
Underbridge Clearance	169 - 171	(3)	0	No Data
			1	Unknown
			2	0 - 5
			3	5 - 10
			4	10 - 50
			5	> 50
			6-7	Not Used
Bypass Conditions (Potential within 2km)	172 - 173	(2)	0	No Data
			1	Easy
			2	Difficult
			3	Impossible
Construction Material	174 - 176	(3)	0	No Data
			1	Other
			2	Wood
			3	Stone/masonry/brick
			4	Steel
			5	Concrete
			6	Reinforced concrete
			7	Prestressed concrete
Classification (one-way wheeled) (metric tons)	177 - 180	(4)	0	No Data
			1	50
			2-9	
			10-15	Not Used
Classification (one-way tracked) (metric tons)	181 - 184	(4)	0	No Data
			1-7	0 - 60 (by 10)
			8	61 - 100
			9	> 100
			10-15	Not Used
Reliability of Bridge Classification	185 - 186	(2)	0	No Data
			1	Unknown
			2	Known
			3	Estimated
Spans (number)	187 - 190	(4)	0	No Data
			1	Unknown
			2	1
			3	2
			4	3
			5-8	4 - 11 (by 2's)
			9	>= 12
			10-15	

Data Element	Bit Designation	# of Bits	Code	Value Represented
Span Length (meters)	191 - 193	(3)	0	No Data
			1	Unknown
			2	< 25
			3	25 - 50
			4	50 - 100
			5	>100
			6-7	

VI. Obstacles Overlay

Type	194 - 196	(3)	0	No Data
			1	Road and RR cuts and fills
(Note: Linear obstacles are defined as any hinderance to movement which is greater than 1.5 meters high, has a 45% or greater slope, and is at least 250 meters long. Areal obstacles are defined as any area which is so characterized as to severely restrict, stop, or otherwise make movement impractical. This overlay depicts land obstacles only-user is referred to Surface Drainage Overlay for hydrologic obstacles.)			2	Natural linear obstacles (escarpments, dikes, cliffs,
			3	Walls and/or fences movement (hedgerows, rock and wire fences and retaining walls, etc.)
			4	Other man-made linear obstacles (dikes, moats, embankments, etc.)
			5	Military obstacles (antitank ditches, airfield and/or road craters, blown bridges, debris choked valleys and/or towns, impact areas, minefield, road-blocks, trenches, wire entanglements, etc.)
			6	Man-made areal obstacles(mining) operations--pits, quarries, strip mines, etc.--terraced hills and/or paddies (wet and dry))
			7	Natural area obstacles (craters, dissected land, talus piles, depressions, sink holes, open water, etc.)
Height (m)	197 - 199	(3)	0	No Data
			1	< 1.5
			2	1.5 - 5
			3	5 - 10
			4	10 - 20
			5	20 - 35
			6	> 35
			7	

The Prototype Fort Lewis-Yakima Training Area TTADB matrixed format ends with bit 199. The full proposed TTADB includes 224 bits and includes additional overlays for:

(1) Aerial Obstructions.

(2) Special Features/Product Synthesis (customer related overlays/standardized and future CCM, Concealment, etc. overlays).

(3) Text Data (two) - for such information as bridge tables, climatic data, hydrologic flow graphs, names, generalized descriptors, etc. On account of the limitations mentioned on page 1, rather than carry 25 bits packed with zeroes at the end of each data point, it was decided to omit these fields from the Fort Lewis-Yakima Training Area TTADB.

APPENDIX E

RESULTS OF FIELD REDUCTION OF THE TACTICAL TERRAIN ANALYSIS DATA BASE (TTADB) FORMAT FIELDS REMAINING

Data Element	Bit Designation	# of Bits	Code	Value Represented
Surface				
Elevation (m)	1 - 16	(16)		
Vegetation Overlay				
Type	21 - 26	(5) vice (6)	0	No Data
			1	Agriculture (dry crops)
			2	Agriculture (wetland rice)
			3	Agriculture (terraced crop: both wet and dry)
			4	Agriculture (shifting cult
			5	Brushland/Scrub (<5m.high, nearly open to medium sp:
			6	Brushland/Scrub (<5m high, medium to dense spacing)
			7	Coniferous/Evergreen Fores
			8	Deciduous Forest
			9	Mixed Forest
			10	Orchard/Plantation (rubber, palm, fruit, etc.)
			11	Grassland, Meadows, Pasture
			12	Grassland with Scattered Trees Some Scrub Growth
			13	Forest Clearings (cutover areas, burns, etc.)
			14	Swamp(mangrove,cypress,etc)
			15	Marsh/Bog(peat, muskeg, etc)
			16	Wetlands (L.S.I., low-lying wet areas)
			17	Vineyard/Hop-garden
			18	Bamboo
			19	Bare Ground
			20	Open Water (was 23)
			21	Built-up Areas(was 24)
Surface Drainage				
Type	96-98	(3)	0	No Data
			1	Stream Channel (Dry Wash or Intermittent, e.g., Arroyo)
			2	Lakes, Ponds, Reservoirs
			3	Stream Channel (Perennial)
			4	Stream Channel (Subject to Tidal fluctuations)
			5	Trenched Stream/Irrigation Canal/Canal/Drainage Ditch
			6	Off-Route Ford (Entrance and Exit points connected)
			7	Dam/Lock

Data Element	Bit Designation	# of Bits	Code	Value Represented
Transportation Overlay				
Type	123 - 126	(4)	0	No Data
			1	Bridge - Road
			2	Bridge - Railroad
			3	Tunnel - Road
			4	Tunnel - Railroad
			5	Dual Lane/Divided Highway/ Expressway
			6	Highway/Road
			7	Railroad
			8	Airfield
			9	Inland Waterway
			10	Lock
Qualifier	130 - 132	(2) vice (3)	0	No Data
			1	Ferry Site
			2	On Route Ford Site
			3	Electrified Line
Highways / Roads:				
Type	147 - 149	(3)	0	No Data
			1	All Weather
			2	Fair/Dry Weather
			3	Cart Tracks
			4	Trails

APPENDIX F

VISTA: VISION, INTERVISIBILITY, SURROGATE TRAVEL, TERRAIN ANALYSIS, ANALYSIS

(This description is made of excerpts from the abstract explaining the VISTA system, written by D. Hoffman, D. H. McCoy, and H.J.de St. Germain, US Army TRADOC Systems Analysis, Activity White Sands Missile Range, New Mexico 88002-5502, USA.)

VISTA is a new interactive terrain analysis graphics program developed by the Armored Systems Division, US Army TRADOC Systems Analysis Activity (TRASANA). VISTA can be used to display and analyze terrain data in a variety of ways. The supporting data base typically consists of elevations, surface features, roads, rivers, hydrography, and obstacles. VISTA currently operates on a 12x12 kilometer area at 100-meter resolution. The terrain data can be displayed in map form in three basic ways: as shaded relief (with variable sun angle), color contours, or regular contours. There are three general modes with which the analyst may examine various aspects of the data: LINE-OF-SIGHT (LOS), PERSPECTIVE VIEWING, and SURROGATE TRAVEL. LOS itself may also be displayed in three different ways. Profile LOS allows the user to select the standard point-to-point line-of-sight which may be used to help determine locations for systems placement (e.g., weapons, sensors, or communications equipment). Masked LOS and Multi-Sensor LOS can be used to determine line-of-sight for various forward observers, surveillance positions, and weapon sites. PERSPECTIVE VIEWING allows the user to position an observer at any point on, or outside the terrain, and to obtain a 3-D view of the terrain and surrounding features as they would appear from that point. SURROGATE TRAVEL allows the user to develop animated 3-D perspective views of travel along defined surface or aerial routes within the terrain area.

1. Introduction

VISTA is a low-cost, near realtime, sophisticated terrain analysis system. The acronym VISTA means Vision, Intervisibility, Surrogate Travel, Terrain Analysis, Analysis and should not be confused with the Army program VISTA, which means Very Intelligent Surveillance and Target Acquisition. The package was developed by the Armored Systems Division of the TRADOC Systems Analysis Activity. Less than one man year of effort has been expended on the initial development of VISTA.

2. Background

Vista was developed as a tool to perform scenario development, terrain analyses to support TRADOC studies, and as an ABCA research project. The number of requests for. and use of VISTA, clearly indicate that this development was indeed a step in the right direction. To our knowledge, VISTA is the most effective tool of its kind developed solely by Army sources without contractor help. It has a variety of uses and operates in a near realtime mode, even for complex perspective scene generation.

3. Vista Capabilities

VISTA may be used to generate two-dimensional map displays using three different landform representations. First, one may display an ordinary contour map; second, color contour banding may be used (eleven color bands are available); third, shaded relief may be generated. Surface features may be added to all three forms. These features include buildings, vegetation, roads, rivers, lakes, and several classes of obstacles.

Three forms of line-of-sight (LOS) analysis may be conducted with VISTA. These include profile LOS, masked LOS, and composite LOS. The profile LOS allows one to pick observer and target locations; VISTA then displays a terrain profile between those points. The intervening features are

exhibited on this profile. Masked LOS is used to examine the LOS from a given point out to a given maximum range. Those areas not seen are masked out in black. Composite LOS is just that, a combination of LOS coverages from two or more sensors using up to eight colors to show the union and intersection of those coverages. Up to forty sensors may be processed, but only the first three will have unique colors to indicate the various coverages.

Perspective views are user-controlled, in that the observer locations, direction, and viewing angle are selectable in the two-dimensional terrain view. The three-dimensional view is then generated and displayed above the two-dimensional. These perspective views may be recorded in series to provide animated travel through the terrain.

4. Vista Hardware/Software Requirements

VISTA runs on a VAX 11/780 computer. The present implementation contains 7500 lines of FORTRAN code. The graphics device used is a RAMTEK 9400 with eight bit planes. The eight bit planes are configured into three overlay planes. Note: The shaded relief presentation could be improved with more planes. The RAMTEK driver software (between the VAX and RAMTEK) was provided by RAMTEK. The graphics library was created locally, after DI 3000 proved to be inadequate for most TRASANA applications.

5. Vista Algorithms and Data

Two LOS algorithms are used in VISTA. One, the so-called DYNTACS algorithm, uses quadratic interpolation to generate elevation values between grid points. The other is a nearest square algorithm called the Bresenham algorithm because it was adapted from Bresenham's straight line pixel lighting algorithm.

The perspective is generated by defining a viewing window in the world coordinate system. The scene is projected into the window using reflected light (a function

of the cosine of the angle between the light source and the normal to the terrain surface). The scene is then scaled to fit the screen coordinates. The scene is completely generated on the VAX, then transferred to the RAMTEK display.

Defense Mapping Agency (DMA) terrain data are used whenever possible. Because of content and resolution, DMA CATTS/ARTBASS data are preferred. DTED Level I is used for ground-to-ground LOS computations as a last resort, due to quality, content, and the WGS coordinate system.

6. Future Plans For VISTA

VISTA is being modified to enhance its capabilities for doing terrain analysis. TRASANA plans to use the system to support a major air defense study (FAAD, forward area air defense) in the immediate future. Statistical packages from other terrain analysis programs will be modified and added as time permits. As hardware improvements become available, it is believed VISTA can be modified to generate full screen perspective views in less than a second, with up to 64 levels of shading. Scene generation using display lists is being investigated on a Chromatics system.

7. VISTA Availability

VISTA is available to all DoD users. Contractors are asked to sign a memorandum of understanding prior to software release. To date, six agencies are using VISTA for terrain analysis and other study support.

To obtain more information about VISTA, contact:

Mr. D. Hue McCoy
US Army TRADOC Systems Analysis Activity
ATTN: ATOR-TFC
White Sands Missile Range
New Mexico 88002-5502
COMM: (505) 678-1551/5891
FTS: 898-1551/5891
AVON: 258-1551/5891

APPENDIX G

TDMS.C PROGRAM LISTING

```
/*****

                    TACTICAL DIGITAL MAPPING SYSTEM

                              PROTOTYPE

                                 by

                    Maj. S. J. Gaffney  and  Capt. J. E. Daly


  This program was written on an IBM PC AT and equipped with
an EGA board and an AT clone with EGA.  Lattice's  'C' Compiler
ver. 3.1 was used in conjunction with the Essential Software's
"C Utility Library",ver. 3.0.  Mouse Systems' PC Paint ver. 1.5
was also used to create the binary .PIC files called into the
program.

  This program is used in conjunction with another program named
CREATE.C to produce the functioning prototype program.

*****/
m
/***** System Variables *****/
#include "colors.h"
#include "ctype.h"
#include "dos.h"
#include "error.h"
#include "filedata.h"
#include "graphics.h"
#include "intreqs.h"
#include "math.h"
#include "stdio.h"

/***** Global Variables *****/
#define BACKGRND  BLUE                  /* Background color */
#define FOREGRND  YELLOW                /* Foreground color */
#define ERRCOLOR  YELLOW                /* Foreground color */
#define FORTY  0                        /* Code for forty column mode */
#define EIGHTY  2                       /* Code for eighty column mode */
#define HIGH  2                         /* High res code */
#define MEDIUM  1                       /* Medium res code */
#define TEXT  0                         /* Text mode code */

long int _stack = 40000;
qvideo();
```

```
/*****************************************************************************
*                                                                            *
*                              MAIN                                          *
*                                                                            *
*****************************************************************************/

main()

{
initgraf(0, 0, 0);                      /* Clear screen and set to hires mode. */
logos();                                /* Starting logos. */
clscolor (FOREGRND, BACKGRND);          /* Clear screen and set colors. */
border (BACKGRND);                      /* Set border color. */
sysmain();                              /* Call the sys_main routine. */

done : initgraf(0, 0, 0);               /* Set screen display to hi-res and set colors. */
return;
}


/*****************************************************************************
*                                                                            *
*                              SYSMAIN                                       *
*                                                                            *
*****************************************************************************/

sysmain ()

{
int key;                                /* Local integer variable. */

clearkbd();                             /* Clear keyboard buffer. */
clscolor (FOREGRND, BACKGRND);          /* Clear screen and set colors. */
page ("MAIN.MNU");                      /* Display the Main Menu screen. */
curlocat (23, 30);                      /* Move cursor to location (23,30). */
menuloop: timeloop();                   /* Display an active date and time. */
ecokey (&key);                          /* Wait for key pressed and print it on screen. */
switch (key)                            /* Set up for menu selections. */

   {
      case '1': intelsys();             /* Call the Intelligence Routine. */
          clscolor (FOREGRND, BACKGRND);/* GO BACK TO MAIN MENU */
        page ("MAIN.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '2': ops_sys();              /* Call the Operations Routine. */
          clscolor (FOREGRND, BACKGRND);/* GO BACK TO MAIN MENU */
        page ("MAIN.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '3': logs_sys();             /* Call the Logistics Routine. */
          clscolor (FOREGRND, BACKGRND);/* GO BACK TO MAIN MENU */
        page ("MAIN.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '4': fire_sys();             /* Call the Fire Support Routine. */
          clscolor (FOREGRND, BACKGRND);/* GO BACK TO MAIN MENU */
        page ("MAIN.MNU");
        curlocat (23, 30);
            goto menuloop;
```

```
      case '5': quit();                    /* EXIT TO DOS? */
          clscolor(FOREGRND, BACKGRND);    /* NO--> GO BACK TO MAIN MENU */
        page ("MAIN.MNU");
        curlocat (23, 30);
            goto menuloop;
      default:  error();
                goto menuloop;
   }
}

/*****************************************************************************
*                                                                            *
*                  INTELLIGENCE SUBSYSTEM ROUTINES                           *
*                                                                            *
*****************************************************************************/

intelsys()

{
int key;                               /* Local integer variable. */

clearkbd();                            /* Clear keyboard buffer. */
clscolor (FOREGRND, BACKGRND);         /* Clear screen and set colors. */
page ("INTEL.MNU");                    /* Display the Intelligence Menu screen. */
curlocat (23, 30);                     /* Move cursor to location (23,31). */
menuloop: timeloop();                  /* Display an active date and time. */
ecokey (&key);                         /* Wait for key pressed and print it on screen. */
switch (key)                           /* Set up for menu selections. */

  {
      case '1': doscmd("create.exe");  /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);/* GO BACK TO INTEL MENU */
        page ("INTEL.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '2': doscmd("create.exe");  /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);/* GO BACK TO INTEL MENU */
        page ("INTEL.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '3': notyet();              /* Call the NotYet Routine. */
          clscolor(FOREGRND, BACKGRND);/* GO BACK TO INTEL MENU */
        page ("INTEL.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '4': notyet();              /* Call the NotYet Routine. */
          clscolor(FOREGRND, BACKGRND);/* GO BACK TO INTEL MENU */
        page ("INTEL.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '5': quit();                /* EXIT TO DOS ? */
          clscolor(FOREGRND, BACKGRND);/* NO--> GO BACK TO INTEL MENU */
        page ("INTEL.MNU");
        curlocat (23, 30);
            goto menuloop;
      default:  error();
                goto menuloop;
   }
}
```

```
/*****************************************************************************
*                                                                           *
*                      OPERATIONS SUBSYSTEM ROUTINES                        *
*                                                                           *
*****************************************************************************/

ops_sys()

{
int key;                                  /* Local integer variable. */

clearkbd();                               /* Clear keyboard buffer. */
clscolor (FOREGRND, BACKGRND);            /* Clear screen and set colors. */
page ("OPS.MNU");                         /* Display the Operations Menu screen. */
curlocat (23, 30);                        /* Move cursor to location (23,31). */
menuloop: timeloop();                     /* Display an active date and time. */
ecokey (&key);                            /* Wait for key pressed and print it on screen. */
switch (key)                              /* Set up for menu selections. */

  {
      case '1':doscmd("create.exe");      /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);   /* GO BACK TO OPS MENU */
        page ("OPS.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '2': doscmd("create.exe");     /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);   /* GO BACK TO OPS MENU */
        page ("OPS.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '3': doscmd("create.exe");     /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);   /* GO BACK TO OPS MENU */
        page ("OPS.MNU");
        curlocat (23, 30);
            goto menuloop;

      case '4': doscmd("create.exe");     /* DISPLAY A MAP */
          clscolor(FOREGRND, BACKGRND);   /* GO BACK TO OPS MENU */
        page ("OPS.MNU");
        curlocat (23, 30);
        goto menuloop;

      case '5': quit();                   /* EXIT TO DOS ? */
          clscolor(FOREGRND, BACKGRND);   /* NO-->GO BACK TO OPS MENU */
        page ("OPS.MNU");
        curlocat (23, 30);
            goto menuloop;
      default:  error();
                goto menuloop;
   }
}

/*****************************************************************************
*                                                                           *
*                      LOGISTICS SUBSYSTEM ROUTINES                         *
*                                                                           *
*****************************************************************************/

logs_sys()

{
int key;                                  /* Local integer variable. */

clearkbd();                               /* Clear keyboard buffer. */
clscolor (FOREGRND, BACKGRND);            /* Clear screen and set colors. */
page ("LOGS.MNU");                        /* Display the Logistics Menu screen. */
```

```
curlocat (23, 30);                      /* Move cursor to location (23,31). */
menuloop: timeloop();                   /* Display an active date and time. */
ecokey (&key);                          /* Wait for key pressed and print it on screen. */
switch (key)                            /* Set up for menu selections. */

  {
     case '1': doscmd("create.exe");    /* DISPLAY A MAP */
         clscolor(FOREGRND, BACKGRND);  /* GO BACK TO LOGS MENU */
       page ("LOGS.MNU");
       curlocat (23, 30);
           goto menuloop;

     case '2': doscmd("create.exe");    /* DISPLAY A MAP */
         clscolor(FOREGRND, BACKGRND);  /* GO BACK TO LOGS MENU */
       page ("LOGS.MNU");
       curlocat (23, 30);
           goto menuloop;

     case '3': doscmd("create.exe");    /* DISPLAY A MAP */
         clscolor(FOREGRND, BACKGRND);  /* GO BACK TO LOGS MENU */
       page ("LOGS.MNU");
       curlocat (23, 30);
           goto menuloop;

     case '4': doscmd("create.exe");    /* DISPLAY A MAP */
         clscolor(FOREGRND, BACKGRND);  /* GO BACK TO LOGS MENU */
       page ("LOGS.MNU");
       curlocat (23, 30);
           goto menuloop;

     case '5': quit();                  /*  EXIT TO DOS? */
         clscolor(FOREGRND, BACKGRND);  /* NO-->GO BACK TO LOGS MENU */
       page ("LOGS.MNU");
       curlocat (23, 30);
           goto menuloop;
     default:  error();
               goto menuloop;
  }
}

/*****************************************************************************
*                                                                            *
*                      FIRE SUPPORT SUBSYSTEM ROUTINES                       *
*                                                                            *
*****************************************************************************/

fire_sys()

{
int key;                                /* Local integer variable. */

clearkbd();                             /* Clear keyboard buffer. */
clscolor (FOREGRND, BACKGRND);          /* Clear screen and set colors. */
page ("FIRE_SUP.MNU");                  /* Display the Fire Support Menu screen. */
curlocat (23, 30);                      /* Move cursor to location (23,31). */
menuloop: timeloop();                   /* Display an active date and time. */
ecokey (&key);                          /* Wait for key pressed and print it on screen. */
switch (key)                            /* Set up for menu selections. */

  {
     case '1': notyet();                /* Call the NotYet Routine. */
         clscolor(FOREGRND, BACKGRND);  /* GO BACK TO FIRE_SUP MENU */
       page ("FIRE_SUP.MNU");
       curlocat (23, 30);
           goto menuloop;
```

```
        case '2': doscmd("create.exe");      /* DISPLAY A MAP */
           clscolor(FOREGRND, BACKGRND);     /* GO BACK TO FIRE_SUP MENU */
          page ("FIRE_SUP.MNU");
          curlocat (23, 30);
             goto menuloop;

        case '3': doscmd("create.exe");      /* DISPLAY A MAP */
           clscolor(FOREGRND, BACKGRND);     /* GO BACK TO FIRE_SUP MENU */
          page ("FIRE_SUP.MNU");
          curlocat (23, 30);
             goto menuloop;

        case '4': doscmd("create.exe");      /* DISPLAY A MAP */
           clscolor(FOREGRND, BACKGRND);     /* GO BACK TO FIRE_SUP MENU */
          page ("FIRE_SUP.MNU");
          curlocat (23, 30);
             goto menuloop;

        case '5': quit();                    /* EXIT TO DOS ? */
           clscolor(FOREGRND, BACKGRND);     /* NO-->GO TO FIRE_SUP MENU */
          page ("FIRE_SUP.MNU");
          curlocat (23, 30);
             goto menuloop;
        default:  error();
                  goto menuloop;
     }
}


/*****************************************************************************
*                                                                            *
*                      SUBSYSTEM UTILITY ROUTINES                            *
*                                                                            *
*****************************************************************************/


/*****************************************************************************
*                                                                            *
*                                FLASH                                       *
*                                                                            *
*****************************************************************************/

/*****
  The flash routine writes a stored graphics file to the screen buffer and displays it on the monitor.

PARAMETERS:
  screen - file name with the graphics
  resolution - screen resolution
  palette - palette used (0 or 1)
  bakgrnd - background/border color
*****/


flash(screen, resolution, palette, bakgrnd)

char *screen[];                          /* Set a pointer to external character screen array. */
int resolution, palette, bakgrnd;        /* External integer variables. */

{
   char buff[16400];                     /* Initialize a character buffer. */
   struct intregs regs;                  /* Local structure variables. */
   unsigned failure, handle, actual;     /* Local unsigned variables. */
```

```
/***** Read in the image. *****/

   if ((failure=openfil(screen,0,&handle)) > 0)
   {
      clscolor (FOREGRND, BACKGRND);        /* Clear screen and set colors. */
      curlocat ( 9, 0);                     /* Start writing at cursor location (9,0). */
      colrprts ("********** FLASH cannot open screen file ********** " ,ERRCOLOR,BACKGRND);
      curlocat (11, 10);                    /* Start writing at cursor location (11,10). */
      colrprts (screen, FOREGRND, BACKGRND);  /* Print screen in set colors. */
      curlocat (13, 10);                    /* Start writing at cursor location (13,10). */
      colrprts ("  Press a key to continue", FOREGRND, BACKGRND);
      return;
   }

   readfil (handle, 16400, buff, &actual); /* Read the binary data into temporary file. */
   closefil (handle);                      /* Close the temporary file. */
   initgraf (resolution, palette, bakgrnd); /* Initialize the graphic screen. */
   qdosint (0, &regs, &regs);              /* Get the segment value. */
   segmov (16376, &buff[7], regs.ds, 0, 0xb800); /* Flash the image into the screen buffer. */
   return;
}


/*****************************************************************************
*                                                                            *
*                               PAGE                                         *
*                                                                            *
*****************************************************************************/

/*****
  Page writes a screen of text (ASCII) to the screen.

PARAMETERS:
  page_file -- name of text file

CONTROL CODES:
   .c n -   Center the next n lines.
   .e -     Echo the output to the printer if the user requests it.
   .m -     Display the centered message: "Press any key to continue or Q to quit"
            on line 24.  Pressing a key causes screen erasure.
   .p -     Pause. Pressing a key causes screen erasure.
   .4 -     Forty column mode (erases screen).
   .8 -     Eighty column mode (erases screen).
*****/

page (page_file)
char *page_file;                        /* Set pointer. */

{
   int cent, col, i, mode, pflag, plines, row;   /* Local integer variables. */
   char *flag;                          /* Set pointer. */
   char line[80];                       /* Open character array of 80. */
   FILE *infile, *fopen();              /* Set pointers. */
   pflag = 0;                           /* Initialize pflag value. */

/****** Open the page file *****/

   if ((infile = fopen (page_file, "r")) == NULL)
   {
      clscolor (FOREGRND, BACKGRND);        /* Clear screen and set colors.  */
      curlocat( 9, 0);                      /* Start writing with cursor at (9,0).  */
      colrprts ("********** PAGE cannot open text file. **********  " , FOREGRND, BACKGRND);
      curlocat (11, 10);                    /* Start writing with cursor at (11,10). */
      colrprts (page_file, FOREGRND, BACKGRND); /* Clear screen and set colors.  */
      curlocat (24, 30);                    /* Start writing with cursor at (23,33). */
      return;
   }
```

```
/***** Begin reading in text. *****/

   row = 0;                                /* Initialize row value. */
   cent = 0;                               /* Initialize cent value. */
   while ((flag = fgets (line, 80, infile)) != NULL)
   {
      if (line[0] == '.')                  /* If 1st line array value = . then ... */
      {

/***** Interpret the command. *****/

         switch (line[1])                  /* Set up command code variable array. */
         {


/***** .c n -   Center the next n lines. *****/

         case 'c':
            cent = atoi (&line[2]);
            break;                         /* Break out of the loop. */


/***** .e -     Echo the output to the printer. *****/

         case 'e':
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors.  */
            curlocat (12, 23);             /* Start writing with cursor at (12, 23). */
            colrprts (" Do you want printed output (Y/N)?", FOREGRND,BACKGRND);
            if (getyesno(1))               /* Get Boolean value of Y/N input. */
            {
               pflag = 1;                  /* Set pflag value. */
               for (i=1; i <= 6; i++) lprtlf();  /* Print line for i 1 to 6 times. */
            }
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors.  */
            plines = 0;                    /* Initialize value. */
            break;                         /* Break out of the loop. */


/***** .m -     Display the centered message: "Press any key to continue or Q to quit"
                on line 24.  Pressing a key causes screen erasure. *****/

         case 'm':
            getscmod (&mode, &i, &i);      /* Get screen display mode integer value. */
            if (mode <= 1) curlocat (24, 2);  /* If Boolean false, start writing at curcor location (24, 2). */
            else curlocat (24, 21);        /* Start writing with cursor at (24, 21). */
            colrprts (" Press any key to continue or Q to quit", FOREGRND, BACKGRND);
            getkey(&i);                    /* Wait for key pressed. */
            if ((i == 'q') || (i == 'Q'))  /* If key pressed. */
            {
               fclose (infile);
               if (pflag)                  /* If pflag set. */
               lprtff();                   /* Print infile. */
               return;
            }
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
            row = 0;                       /* Initialize row value. */
            break;                         /* Break out of the loop. */


/***** .p -     Pause. *****/

         case 'p':
            pause();                       /* Wait for key pressed. */
            row = 0;                       /* Initialize row value. */
            break;                         /* Break out of the loop. */
```

```
/***** .4 -     Forty column mode (erases screen). *****/

        case '4':
           setscmod (FORTY);              /* Set screen display mode to 40 columns. */
           border (BACKGRND);             /* Set border color. */
           clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
           row = 0;                       /* Initialize row value. */
           break;                         /* Break out of the loop. */


/***** .8 -     Eighty column mode (erases screen). *****/

        case '8':
           setscmod (EIGHTY);             /* Set screen display mode to 80 columns. */
           border (BACKGRND);             /* Set border color. */
           clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors.  */
           row = 0;                       /* Initialize row value. */
           break;                         /* Break out of the loop. */
        }
     }
     else
     {
         line [strlen (line) -1] = '\0';  /* Print the line. */
         if (cent > 0)
     {

/***** Process centered lines. *****/

       getscmod (&mode, &i, &i);          /* Determine display mode. */
           --cent;                        /* Decrement cent. */
           if (mode <= 1) col = (40 - strlen (line)) /2;  /* Calculate center of line if 40 col. mode. */
           else col = (80 - strlen (line)) /2;  /* Calculate center of line if 80 col. mode. */
        }
        else col = 1;                     /* Set col = 1. */
        if (row >= 24)
        {
           row = 0;                       /* Initialize row. */
        }
        if (pflag && (plines > 54))       /* If pflag set and number of lines > 54. */
        {
           plines = 0;                    /* Initialize plines. */
           lprtff();                      /* Print line. */
           for (i=1; i <= 6; i++) lprtlf();  /* Increment i from 1 to 6. */
        }
        if (pflag)                        /* If pflag set. */
        {
           if (col > 1) for (i=1; i < col; i++) lprtchar(0, ' ');
           lprtstr (line);                /* Print string. */
           lprtcr();
           lprtlf();                      /* Print line. */
           plines++;                      /* Increment plines. */
        }
        curlocat (++row, col);            /* Start writing with cursor at (row,col). */
        colrprts (line, FOREGRND, BACKGRND);  /* Clear screen and set colors.  */
     }
  }

  fclose (infile);                        /* Close file buffer. */
  if (pflag)                              /* If pflag set. */
  lprtff();                               /* Print infile. */
  return;
}
```

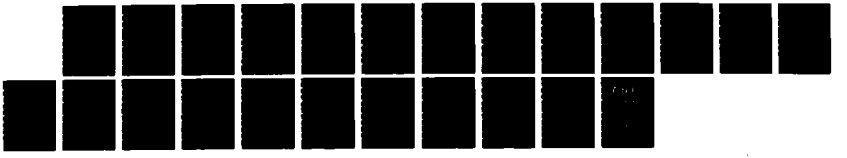
```
/*****************************************************************************
*                                                                           *
*                                TIMELOOP                                   *
*                                                                           *
*****************************************************************************/

/*****
  Timeloop displays the system time and date at the bottom of the menu title board.

PARAMETERS:
  None.
*****/

timeloop()

{
   int i, j, row, col;                    /* Local integer variables. */
   char dt[30],                           /* Date display buffer. */
        tm[12];                           /* Time display buffer. */

   clearkbd();                            /* Clear keyboard buffer. */
   curget(&row, &col);                    /* Place cursor at location of pointers (row,col). */
   for( ; ; )                             /* Start infinite loop. */

   {
      curtype (1, 0, 0);                  /* Turn cursor off. */
      if((i = checkkey()) == 1) break;    /* Wait for key pressed. */
      systime (&tm, 9);                   /* Load the systime buffer in the #9 format. */
      sysdate (&dt, 24);                  /* Load the sysdate buffer in the #24 format. */
      curlocat ( 8,18);                   /* Start writing at cursor location (8,12). */
      colrprts(dt,FOREGRND,BACKGRND);     /* Print contents of the sysdate buffer in set colors. */
      curlocat( 8,52);                    /* Start writing at cursor location (8,46). */
      colrprts (tm, FOREGRND, BACKGRND);  /* Print contents of the systime buffer in set colors. */
      curlocat(row,col);                  /* Start writing at cursor location (row,col). */
      curtype(0,6,7);                     /* Turn cursor on and set colors. */
      for (i = 1; i <= 1000; i++)         /* Increment i from 1 to 1000. */

      {
         if((j = checkkey()) == 1) goto done;  /* On key pressed log off and exit to DOS. */
      }
   }
   done:  curtype(0,6,7);                 /* Turn cursor on and set colors. */
   return;
}



/*****************************************************************************
*                                                                           *
*                              TIMEWINDOW                                   *
*                                                                           *
*****************************************************************************/

/*****
  Timewindow displays the system time and date in the bottom right corner of the screen when control
  t is pressed.  Control t again turns it off.

PARAMETERS:
  None.
*****/

timewindow()

{
   int i, j;                              /* Local integer variables. */
   char time[16],                         /* Time display buffer. */
        tm[8];
```

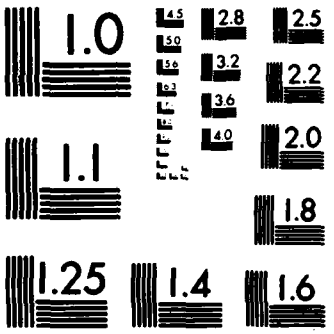
```
   for( ; ; )                              /* Start infinite loop. */
   {
      systime (&time, 9);                  /* Load the systime buffer in the 89 format. */
      systime (&ta, 9);                    /* Load the systime buffer in the 89 format. */
      swpwindo (24, 72, 24, 79, time);     /* Open window in bottom right corner of screen. */
      curlocat(24,72);                     /* Start writing at cursor location (24,72). */
      colrprts (ta, FOREGRND, BACKGRND);   /* Print contents of the systime buffer in set colors. */
      for (i = 1; i <= 1000; i++)          /* Increment i from 1 to 1000. */

      {
         if(j = checkkey()) = 1) goto done;  /* On key pressed log off and exit to DOS. */
      }

   }
   done:  curtype(0,6,7);                  /* Turn cursor on and set colors. */
   swpwindo(24, 72, 24, 79, time);         /* Close window in bottom right corner of screen. */

   return;
}



/*****************************************************************************
*                                                                            *
*                               HEAD                                         *
*                                                                            *
*****************************************************************************/

/*****
  The head routine clears the screen and prints the character
  string in title centered on line 1.  Eighty column mode is used.

PARAMETERS:
  None.
*****/

head(title)

char *title;                               /* Pointer to external character variable. */

{
   int i;                                  /* Local variable. */

   clscolor(FOREGRND,BACKGRND);            /* Clear screen and set colors. */
   i = (80 - strlen (title)) /2;           /* Calculate center of the line. */
   curlocat(1,i);                          /* Start writing at cursor location (1,i). */
   colrprts(title,FOREGRND,BACKGRND);      /* Print title in set colors. */
   return;
}



/*****************************************************************************
*                                                                            *
*                               LOGOS                                        *
*                                                                            *
*****************************************************************************/

/*****
   The logos routine displays the logos at the beginning of the dialog.

PARAMETERS:
  None:
*****/
```

```
logos()
{
   initgraf(0,0,0);                          /* Set display to hires mode, black on black. */
   clearkbd();                               /* Clear keyboard buffer. */
   flash("TITLE.PIC",MEDIUM,0,BLACK);        /* Call flash to display TITLE. */
   syspsany(0,0,5,0);                        /* Wait for five seconds or key pressed. */
   initgraf(0,0,0);                          /* Set display to hires mode, black on black. */
}


/*****************************************************************************
*                                                                            *
*                                 PAUSE                                      *
*                                                                            *
*****************************************************************************/

/*****
  Pause causes a pause until a key is pressed.

PARAMETERS:
  None.
*****/

pause()
{
   int i;                                    /* Local Boolean variable. */

   clearkbd();                               /* Clear keyboard buffer. */
   for ( ; ; )                               /* Starts an infinite loop. */

   {
      if((i = checkkey()) == 1) break;       /* Poll keyboard for an input. */
   }

   clearkbd();                               /* Clear keyboard buffer. */
   clscolor (FOREGRND, BACKGRND);            /* Clear screen and set colors. */
}


/*****************************************************************************
*                                                                            *
*                                 NOTYET                                     *
*                                                                            *
*****************************************************************************/

/*****
  Notyet is a dummy routine to hold a place for future functions.

PARAMETERS:
  None.

MENU:


               THIS FUNCTION NOT YET IMPLEMENTED




      Press a key to continue.

*****/
```

AD-A186 281 TACTICAL USE OF DIGITIZED MAPS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S J GAFFNEY ET AL SEP 87 2/2

UNCLASSIFIED F/G 8/2 NL





```
notyet()

{
   int i;                                  /* Local integer variable. */
   clscolor(FOREGRND,BACKGRND);            /* Clear screen and set colors. */
   page ("NOTYET.MNU");                    /* Display the notyet message. */
   curtype (0, 0, 0);                      /* Turn off cursor. */
   curlocat (24, 1);                       /* Set cursor at location (24,1). */
   getkey(&i);                             /* Wait for key pressed. */
   curtype (0, 0, 0);                      /* Turn on cursor. */
   sysmain();                              /* Return to sysmain routine. */
}

/*****************************************************************************
*                                                                            *
*                                ERROR                                       *
*                                                                            *
*****************************************************************************/

/*****
  The error routine beeps the speaker and returns the cursor to the same column, previous row.
  It is used to signal bad data input.

PARAMETERS:
  None.
*****/

error()

{
   int row, col;                           /* Initialize row, col integer variables. */
   curget (&row, &col);                    /* Set location of cursor at pointers row, col. */
   curlocat (row, col - 1);                /* Start writing at cursor location (9,2). */
   beep();                                 /* Sound the error warning. */
   return;
}


/*****************************************************************************
*                                                                            *
*                                ERRMSG                                      *
*                                                                            *
*****************************************************************************/

/*****
  This routine prints a centered error message on lineno, beeps once and pauses.
  The user strikes a key, the message is cleared and control is returned.

PARAMETERS:
  lineno -- Location of the cursor and the error message.
  msg    --  Error message to be printed.
*****/

errmsg (lineno, msg)

int lineno;                                /* External integer variable. */
char *msg;                                 /* Pointer to external character variable. */

{
   int i, mode;                            /* Local integer variables. */
   getscmod (&mode, &i, &i);               /* Determine screen display mode and set colors. */
   if (mode) i = (80 - strlen (msg)) /2;   /* Calculate center of string in 80 column mode. */
   else i = (40 - strlen (msg)) /2;        /* Calculate center of string in 40 column mode. */
   curlocat(lineno, i);                    /* Start writing at cursor location (9,2). */
   colrprts(msg, ERRCOLOR,BACKGRND);       /* Print in set colors. */
   beep();                                 /* Sound error warning. */
   getkey (&i);                            /* Wait for key pressed. */
   curlocat(lineno, i);                    /* Start writing at cursor location (9,2). */
```

```
   colrprts(
                        FOREGRND,BACKGRND);  /* Print in set colors. */
}

/*****************************************************************************
*                                                                            *
*                               QUIT                                         *
*                                                                            *
*****************************************************************************/

/*****
  Quit causes the program to  exit to DOS.

PARAMETERS:
  None.

MENU:


              YOU ARE ABOUT TO EXIT TO DOS !!!



                  Quit Y/N? ___

*****/

quit()
{
   int i;                               /* Local integer variables. */

   clscolor (FOREGRND, BACKGRND);       /* Clear screen and set colors. */
   clearkbd();                          /* Clear keyboard buffer. */
   page ("QUIT.MNU");                   /* Display the Exit Menu. */
   curlocat (24, 30);                   /* Set cursor at location (24,30). */
   timewindow();                        /* Puts time in lower right corner of screen in reverse video. */

      i= ecoyesno(24, 30, 1);           /* Check keyboard for a Y/N input with echo. */
      if (i == 1)
      {
         initgraf (0, 0, 0);            /* Clear screen. */
         exit();                        /* Exit to DOS. */
      }
      else
      sysmain();                        /* Return to sysmain routine. */
}
```

APPENDIX H

CREATE.C PROGRAM LISTING

```
/*****

                    TACTICAL DIGITAL MAPPING SYSTEM

                              PROTOTYPE

                                 by

                    Maj. S. J. Gaffney  and  Capt. J. E. Daly

      This program is required for operation of the TDMS.C file
    to produce the prototype program.

    /*****



#include "colors.h"
#include "ctype.h"
#include "dos.h"
#include "error.h"
#include "filedata.h"
#include "graphics.h"
#include "intregs.h"
#include "math.h"
#include "stdio.h"


/***** Global Variables *****/

#define BACKGRND  BLUE                  /* Background color */
#define FOREGRND  YELLOW                /* Foreground color */
#define ERRCOLOR  YELLOW                /* Foreground color */
#define FORTY  0                        /* Code for forty column mode */
#define EIGHTY  2                       /* Code for eighty column mode */
#define HIGH  2                         /* High res code */
#define MEDIUM  1                       /* Medium res code */
#define TEXT  0                         /* Text mode code */

long int _stack = 40000;
qvideo();

/*****************************************************************************
*                                                                            *
*                              MAIN                                          *
*                                                                            *
*****************************************************************************/

main()
{
  int  key, i, j, k;
  char storegrd[5], neighbor[8][5], gridfle[8][10], mapdata[10],
     gridstr[5], *mapstore[170];
  FILE  *outfile, *afile, *bfile, *cfile, *dfile,*fopen();

  initgraf(0,0,0);
  clearkbd();                           /* Clear keyboard buffer. */
  clrscrn();                            /* Clear screen  */
  page ("GETMAP.MNU");                  /* Display the getmap Menu screen. */
```

```
curlocat (23, 30);                      /* Move cursor to location (23,31). */
menuloop: timeloop();                   /* Display an active date and time. */
ecokey (&key);
switch (key)

 {
 case '1' : clrscrn();                  /* CASE OF OLD MAP */
            page ("GETGRID.MNU");
            curlocat ( 23, 30);
          getcstr(4,0,0,1,2,*gridstr,0);  /* GET INPUT */
            strcpy(mapdata, "M");       /* CREATE MAP FILENAME */
          strcat(mapdata,*gridstr);
          strcat(mapdata,".DAT");
          if (filexist(&mapdata)== 0)   /* CHECK IF IT EXISTS */

           {
           clscolor(FOREGRND,BACKGRND);
           curlocat( 9, 21);
           colrprts("*****Cannot find old file ",FOREGRND,BACKGRND);
           curlocat( 11, 30);
           pause();
           page ("GETMAP.MNU");
           curlocat (23, 30);
           }

          else
           {
           clscolor(FOREGRND,BACKGRND);
           prntmap(&mapdata);
           page ("GETMAP.MNU");
           curlocat (23, 30);
           }
          goto menuloop;

   case '2' : clrscrn();                /* CASE OF NEW MAP */
              page("GETGRID.MNU");
          curlocat (23, 29);
          break;

   case '3' :  quit();                  /* QUIT CASE */
          clrscrn();
          page ("GETMAP.MNU");          /* RESULT OF NO ANSWER TO QUIT */
          curlocat (23, 30);
          goto menuloop;

   default  :  error();                 /* BAD INPUT */
          clrscrn();
          page ("GETMAP.MNU");          /* RESULT OF NO ANSWER TO QUIT */
          curlocat (23, 30);
          goto menuloop;
   }

/*****************************************************************/
/*  THIS SECTION DETERMINES THE NEIGHBORING DATA FILES           */
/*    BASED ON THE USER INPUT OF THE LOWER LEFT HAND GRID.       */
/*                                                               */
/*  NEIGHBORS  A/+1    B/+101  C/+201  D/+301                    */
/*                                                               */
/*         E      F/+100  G/+200  H/+300                         */
/*                                                               */
/*****************************************************************/

  getcstr(4,0,0,1,2,*gridstr,0);
  strcpy(storegrd, *gridstr);
  strcpy(neighbor[4], *gridstr);        /* load neighbor E */
```

```
for(i= 5; i(= 7; ++i)                    /* LOAD NEIGHBORS F, G, H */
   {                                     /* BY ADDING 100s */
   if (storegrd[1] == '9')

    {
    storegrd[1] = '0';
    if (storegrd[0] == '9')
       storegrd[0] = '0';
    else
       storegrd[0] = storegrd[0] + 1;
    }

   else
    storegrd[1] = storegrd[1] + 1;
   strcpy( neighbor[i], storegrd);
   }

  strcpy(storegrd, *gridstr);            /* RESET BUFFER */
  if (storegrd[3] == '9')                /* LOAD NEIGHBOR A */
   {                                     /* BY ADDING 1 */
   storegrd[3] = '0';
   if (storegrd[2] == '9')
    storegrd[2] = '0';
   else
    storegrd[2] = storegrd[2] + 1;
   }

  else
   storegrd[3] = storegrd[3] + 1;
  strcpy( neighbor[0], storegrd);

for(i= 1; i(= 3; ++i)                    /* LOAD NEIGHBORS B, C, D */
  {                                      /* BY ADDING 100s */
  if (storegrd[1] == '9')
   {
   storegrd[1] = '0';
   if (storegrd[0] == '9')
    storegrd[0] = '0';
   else
    storegrd[0] = storegrd[0] + 1;
   }

  else
   storegrd[1] = storegrd[1] + 1;
  strcpy( neighbor[i], storegrd);
  }

/*****************************************************************/
/*  THIS SECTION CREATES THE DATA FILE NAMES AND THE             */
/*    COMPOSITE FILE NAME.                                       */
/*                                                               */
/*****************************************************************/

  for ( i = 0; i(= 7; ++i)
   {
   strcpy(gridfle[i], "D");              /* CREATE NEIGHBOR DATA FILENAMES */
   strcat(gridfle[i],neighbor[i]);
   strcat(gridfle[i],".DAT");
   }                                     /* end of for */

  clscolor(FOREGRND,BACKGRND);
  curlocat( 10, 21);
  colrprts("     STANDBY PLEASE",FOREGRND,BACKGRND);
  curlocat( 11, 30);

  strcpy(&mapdata, gridfle[4]);          /* CREATE COMPOSITE MAP FILENAME */
  strfill('M', &mapdata,0,1,10);
```

```
/*****************************************************************/
/*  THIS SECTION OPENS THE OUTPUT AND DATA FILES. THEN IT        */
/*    READS FROM THE DATA FILES AND WRITES TO THE COMPOSITE      */
/*  FILE. A THROUGH D ARE READ FIRST AS A GROUP FROM TOP         */
/*  TO BOTTOM, ROW BY ROW, THEN E THROUGH F.                     */
/*              A B C  D  =) COMPOSITE                           */
/*              E F  G H                                         */
/*                                                               */
/*****************************************************************/

 outfile = fopen(mapdata, "w");          /* OPEN OUTPUT FILE */

 j = 0;
 for ( k = 0; k<= 1; ++k)                /* DO FILES 0-3 THEN 4 - 7 */

  /* THE USE OF SUBSTITUTE DUMMY FILES WILL BE USED IN FUTURE UPDATES*/
  /* TO FILL IN WHERE DATA IS NOT AVAILABLE                         */

  {                                      /* OPEN NEIGHBORS  */
    if (filexist(&gridfle[0+j])== 0)     /* CHECK IF IT EXISTS */

   {
   clscolor(FOREGRND,BACKGRND);
   curlocat( 9, 21);
   colrprts("*************DATA NOT AVAILABLE  ",FOREGRND,BACKGRND);
   curlocat( 11, 30);
   pause();
   page ("GETMAP.MNU");
   curlocat (23, 30);
   goto menuloop;
   }

  else
   afile =  fopen(gridfle[0+j], "r");    /* OPEN DATA FILE */
   if (filexist(&gridfle[1+j])== 0)      /* CHECK IF IT EXISTS */

   {
   clscolor(FOREGRND,BACKGRND);
   curlocat( 9, 21);
   colrprts("*************DATA NOT AVAILABLE  ",FOREGRND,BACKGRND);
   curlocat( 11, 30);
   pause();
   page ("GETMAP.MNU");
   curlocat (23, 30);
   goto menuloop;
   }

  else
   bfile =  fopen(gridfle[1+j], "r");    /* OPEN DATA FILE */
   if (filexist(&gridfle[2+j])== 0)      /* CHECK IF IT EXISTS */

   {
   clscolor(FOREGRND,BACKGRND);
   curlocat( 9, 21);
   colrprts("*************DATA NOT AVAILABLE  ",FOREGRND,BACKGRND);
   curlocat( 11, 30);
   pause();
   page ("GETMAP.MNU");
   curlocat (23, 30);
   goto menuloop;
   }

    else
       cfile =  fopen(gridfle[2+j], "r"); /* OPEN DATA FILE */
       if (filexist(&gridfle[3+j])== 0)   /* CHECK IF IT EXISTS */
```

```
    {
    clscolor(FOREGRND,BACKGRND);
    curlocat( 9, 21);
    colrprts("************DATA NOT AVAILABLE  ",FOREGRND,BACKGRND);
    curlocat( 11, 30);
    pause();
    page ("GETMAP.MNU");
    curlocat (23, 30);
    goto menuloop;
    }

   else
    dfile =  fopen(gridfle[3*j], "r");     /* OPEN DATA FILE */
    for (i = 1; i<=20; ++i)                /* COPY DATA FILES COMPOSITE MAP FILE */

    {
       fscanf(afile,"%160s",mapstore);
       fprintf(outfile,"%160s", mapstore);
       fscanf(bfile,"%160s",mapstore);
       fprintf(outfile,"%160s", mapstore);
       fscanf(cfile,"%160s",mapstore);
       fprintf(outfile,"%160s", mapstore);
       fscanf(dfile,"%160s",mapstore);
       fprintf(outfile,"%160s\n", mapstore);
    }

   fclose(afile);
   fclose(bfile);
   fclose(cfile);
   fclose(dfile);
   j = 4;
   }                                       /* END READ\WRITE FOR */

  fclose(outfile);                         /* COMPOSITE FILE COMPLETE, DONE WITH NEIGHBORS */
  clscolor(FOREGRND,BACKGRND);
  prntmap(&mapdata);
  page ("GETMAP.MNU");
  curlocat (23, 30);
  goto menuloop;

}

/*****************************************************************************/
/*                                                                           */
/*                        SUB-SYSTEM MODULES                                 */
/*                                                                           */
/*****************************************************************************/


/*****************************************************************************/
/*                                                                           */
/*                              PRNTMAP                                      */
/*  Given an 8 grid map file this procedure produces a map on the screen     */
/*                                                                           */
/*****************************************************************************/

prntmap(mapdata)
char *mapdata;

{
  int   elevcolr, descolr, rows, linecnt, gridcnt, i,
    elev, scalebase,   rowstart, rowchng, water;
  char  design, mapsort[6], strelev[4];
  FILE  *infile, *fopen();

  initgraf(0,0,0);
  clearkbd();                              /* Clear keyboard buffer. */
  clrscrn();                               /* Clear screen  */
```

```
    infile =  fopen(mapdata, "r");
    clscolor(FOREGRND,BACKGRND);
    curlocat(0,0);
    elevcolr = GREEN;                       /* INITIALIZE COLORS */
    descolr = BLACK;
    rows    = 1;                            /* INITIALIZE ROW  COUNT */
    gridcnt = 1;                            /* INITIALIZE GRID COUNT */
    linecnt = 1;                            /* INITIALIZE LINE COUNT */
    rowchng = 0;                            /* INITIALIZE ROW CHANGE COUNT */

  for (i=1; i<=1920; ++i)
    {
    fscanf(infile,"%3s",&strelev);
    fscanf(infile,"%5s",mapsort);
    if (i == 1 )                            /* DETERMINE COLOR */

     {
     sscanf(strelev,"%d", &scalebase);
     rowstart= scalebase;                   /* SAVE VALUE OF 1st ROW ELEV */
     }

    if ( rowchng ) 0 )                      /* NEW ROW ADJUST FOR COLOR & SCALE */
     while ( rowchng ) 0)
      {
      --scalebase;
      if (elevcolr >= CYAN )                /* CHANGE COLORS */
         --elevcolr;
      else
         elevcolr = WHITE;
      --rowchng;
      }

    else if ( rowchng < 0)
     while ( rowchng < 0)
      {
      ++scalebase;
      if (elevcolr <= BROWN )               /* CHANGE COLOR */
         ++elevcolr;
      else
         elevcolr = GREEN;
      ++rowchng;
      }

    sscanf(strelev,"%d", &elev);
    if ( elev ) scalebase )                 /* IS ELEV HIGHER ? */
     while ( elev ) scalebase)              /* INCREMENT SCALE TO HIGHER ELEV */

      {
      ++scalebase;
      if (elevcolr <= BROWN )               /* CHANGE COLOR */
         ++elevcolr;
      else
         elevcolr = GREEN;
      }

    else if ( elev < scalebase )            /* IS ELEVATION LOWER ? */
     while ( elev < scalebase )             /* DECREMENT SCALE */

      {
      -- scalebase;
      if (elevcolr >= CYAN )                /* CHANGE COLORS */
         --elevcolr;
      else
         elevcolr = WHITE;
     }                                      /* END CHECK FOR ELEVATION COLOR CHANGE */
```

```
switch (mapsort[1])

 {
 case '0' :
     switch ( mapsort[2] )              /* VEGETATION SORT */

      {
      case '0' :                        /* CLEARING OR NO DATA, NO CHANGE */

      case 'C' :
            design = ' ';
            break;

      case '2' : water = BLUE;          /* WATER , BLUE */
            design = ' ';
            break;

      case 'F' : design = '^';          /* FOREST */
            break;

      case 'S' : design = '`';          /* SWAMP OR WET RICE */
            break;

      case 'B' : design = '*';          /* BRUSH */
            break;

      case 'U' : design = '#';          /* BUILT UP AREA */
            break;

      case 'O' : design = 'o';          /* ORCHARD */
            break;

      case 'T' : design = 't';          /* TERRACED */
            break;

      case '1' : design = '1';          /* DRY STREAMBED */
            break;

      case '3' : design = '1';          /* STREAM CHANNEL */
            water = BLUE;
            break;

      case '6' : design = '}';          /* OFF ROUTE FORD */
            break;
      default :                         /* TREAT AS NO DATA */
            design = ' ';
            break;
      }                                 /* END OF NO DATA/ VEG  SWITCH */
       break;

case 'B' : design = 'B';                /* ROAD BRIDGE */
        break;

case 'G' : design = 'b';                /* RAIL BRIDGE */
       break;

case 'T' : design = '[';                /* ROAD TUNNEL */
       break;

case 'N' : design = ']';                /* RAIL TUNNEL */
       break;

case 'D' : design = '=';                /* DIVIDED, DUAL LANE */
       break;

case 'H' : design = '_';                /* HIGHWAY */
       break;
```

```
case 'R' : design = '+';                /* RAILROAD */
      break;

case 'A' : design = '8';                /* AIRFIELD */
      break;

case 'W' : design = '1';                /* WATERWAY */
      water = BLUE;
      break;

case 'L' : design = 'L';                /* LOCK */
      water = BLUE;
      break;

case 'F' : design = 'F';                /* FERRY SITE */
      water = BLUE;
      break;

case 'S' : design = '}';                /* FORD SITE */
      water = BLUE;
      break;

case 'E' : design = '?';                /* ELECTRIC LINES */
      break;

case 'C' : design = '-';                /* ROAD (ALL WEATHER)*/
      break;

case 'X' : design = '_';                /* ROAD (DRY) */
      break;

case 'Y' : design = ':';                /* CART TRAIL */
      break;

case 'Z' : design = '.';                /* TRAIL */
      break;
default :                               /* TREAT AS NO DATA */
   switch ( mapsort[2] )                /* VEGETATION SORT */

    {
    case '0' :                          /* CLEARING OR NO DATA, NO CHANGE */

    case 'C' :
          design = ' ';
          break;

    case '2' : water = BLUE;            /* WATER , BLUE */
          design = ' ';
          break;

    case 'F' : design = '^';            /* FOREST */
          break;

    case 'S' : design = '"';            /* SWAMP OR WET RICE */
          break;

    case 'B' : design = '*';            /* BRUSH */
          break;

    case 'U' : design = '#';            /* BUILT UP AREA */
          break;

    case 'O' : design = 'o';            /* ORCHARD */
          break;

    case 'T' : design = 't';            /* TERRACED */
          break;
```

```
        case '1' : design = '1';          /* DRY STREAMBED */
              break;

        case '3' : design = '1';          /* STREAM CHANNEL */
              water = BLUE;
              break;

        case '6' : design = '{';          /* OFF ROUTE FORD */
              water = BLUE;
              break;

        default :                         /* TREAT AS NO DATA */
              design = ' ';
              break;
        }                                 /* END OF NO DATA/VEG SWITCH */

       break;
    }                                     /* END OF TRANSPORTATION SWITCH */

   if (rows == 20)                        /* DETERMINE VERTICAL GRID LINES */
    {
    rows=1;
    if (design == ' ')
      design = '|';
    if (gridcnt==4)                       /* END OF LINE ? */

     {
     gridcnt = 1;
     rowchng = scalebase - rowstart;      /* DETERMINE CHANGE FROM START */
     ++linecnt;
     }

    else
     ++gridcnt;
    }

   else
    ++rows;
   if (linecnt == 20)                     /* BOTTOM OF GRIDS ? */
    {
    if (design == ' ')
      design = '_';
    }

   if( water == BLUE)
    {
    colrprta(design,descolr, water);
    water = BLACK;
    }

   else
    colrprta (design, descolr, elevcolr);
   }                                      /* END CREATING PIC FOR LOOP*/

  fclose(infile);
pause();
}                                         /* END OF MODULE */


/*************************************************************************/
/*                                                                       */
/*                        SYSTEM UTILITIES                               */
/*                                                                       */
/*************************************************************************/
```

```
/*************************************************************************/
/*                                                                       */
/*                              PAUSE                                    */
/*                                                                       */
/*************************************************************************/

pause()

{
   int i;                               /* Local Boolean variable. */

   clearkbd();                          /* Clear keyboard buffer. */
   for ( ; ; )                          /* Starts an infinite loop. */

   {
      if((i = checkkey()) == 1) break;  /* Poll keyboard for an input. */
   }

   clearkbd();                          /* Clear keyboard buffer. */
   clscolor (FOREGRND, BACKGRND);       /* Clear screen and set colors. */
}


/*************************************************************************
*                                                                        *
*                              PAGE                                      *
*                                                                        *
*************************************************************************/

/*****
  Page writes a screen of text (ASCII) to the screen.

PARAMETERS:
  page_file -- name of text file

CONTROL CODES:
   .c n -   Center the next n lines.
   .e -     Echo the output to the printer if the user requests it.
   .m -     Display the centered message: "Press any key to continue or Q
              to quit" on line 24.  Pressing a key causes screen erasure.
   .p -     Pause. Pressing a key causes screen erasure.
   .4 -     Forty column mode (erases screen).
   .8 -     Eighty column mode (erases screen).

*****/

page (page_file)
char *page_file;                        /* Set pointer. */

{
   int cent, col, i, mode, pflag, plines, row; /* Local integer variables. */
   char *flag;                          /* Set pointer. */
   char line[80];                       /* Open character array of 80. */
   FILE *infile, *fopen();              /* Set pointers. */
   pflag = 0;                           /* Initialize pflag value. */

/******  Open the page file *****/

   if ((infile = fopen (page_file, "r")) == NULL)
   {
      clscolor (FOREGRND, BACKGRND);    /* Clear screen and set colors. */
      curlocat( 9, 0);                  /* Start writing with cursor at (9,0). */
      colrprts ("********** PAGE cannot open text file. **********  ", FOREGRND, BACKGRND);
      curlocat (11, 10);                /* Start writing with cursor at (11,10). */
      colrprts (page_file, FOREGRND, BACKGRND); /* Clear screen and set colors.  */
      curlocat (24, 30);                /* Start writing with cursor at (23,33). */
      return;
   }
```

```
/***** Begin reading in text. *****/

   row = 0;                              /* Initialize row value. */
   cent = 0;                             /* Initialize cent value. */
   while ((flag = fgets (line, 80, infile)) != NULL)
   {
      if (line[0] == '.')                /* If 1st line array value = . then ... */
      {

/***** Interpret the command. *****/

         switch (line[1])                /* Set up command code variable array. */
         {


/***** .c n -   Center the next n lines. *****/

         case 'c':
            cent = atoi (&line[2]);
            break;                       /* Break out of the loop. */


/***** .e -     Echo the output to the printer. *****/

         case 'e':
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
            curlocat (12, 23);           /* Start writing with cursor at (12, 23). */
            colrprts (" Do you want printed output (Y/N)?", FOREGRND,BACKGRND);
            if (getyesno(1))             /* Get Boolean value of Y/N input. */
            {
               pflag = 1;                /* Set pflag value. */
               for (i=1; i <= 6; i++) lprtlf();  /* Print line for i 1 to 6 X. */
            }
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
            plines = 0;                  /* Initialize value. */
            break;                       /* Break out of the loop. */


/***** .m -     Display the centered message: "Press any key to continue or Q to quit"
                on line 24.  Pressing a key causes screen erasure. *****/

         case 'm':
            getscmod (&mode, &i, &i);    /* Get screen display mode integer value. */
            if (mode != 1) curlocat (24, 2); /* If Boolean false, start writing */
                                         /* at curcor location (24, 2). */
            else curlocat (24, 21);      /* Start writing with cursor at (24, 21). */
            colrprts (" Press any key to continue or Q to quit", FOREGRND, BACKGRND);
            getkey(&i);                  /* Wait for key pressed. */
            if ((i == 'q') || (i == 'Q'))  /* If key pressed. */
            {
               fclose (infile);
               if (pflag)                /* If pflag set. */
               lprtff();                 /* Print infile. */
               return;
            }
            clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
            row = 0;                     /* Initialize row value. */
            break;                       /* Break out of the loop. */


/***** .p -     Pause. *****/

         case 'p':
            pause();                     /* Wait for key pressed. */
            row = 0;                     /* Initialize row value. */
            break;                       /* Break out of the loop. */
```

```
/***** .4 -     Forty column mode (erases screen). *****/

        case '4':
           setscmod (FORTY);              /* Set screen display mode to 40 columns. */
           border (BACKGRND);             /* Set border color. */
           clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
           row = 0;                       /* Initialize row value. */
           break;                         /* Break out of the loop. */


/***** .8 -     Eighty column mode (erases screen). *****/

        case '8':
           setscmod (EIGHTY);             /* Set screen display mode to 80 columns. */
           border (BACKGRND);             /* Set border color. */
           clscolor (FOREGRND, BACKGRND); /* Clear screen and set colors. */
           row = 0;                       /* Initialize row value. */
           break;                         /* Break out of the loop. */
        }
     }
     else
     {
         line [strlen (line) -1] = '\0';  /* Print the line. */
         if (cent ) 0)
     {

/***** Process centered lines. *****/

      getscmod (&mode, &i, &i);           /* Determine display mode. */
          --cent;                         /* Decrement cent. */
          if (mode (= 1) col = (40 - strlen (line)) /2; /* Calculate center */
                                          /*    of line if 40 col. mode. */
          else col = (80 - strlen (line)) /2; /* Calculate center of line */
                                          /*      if 80 col. mode. */
        }
        else col = 1;                     /* Set col = 1. */
        if (row )= 24)
        {
           row = 0;                       /* Initialize row. */
        }
        if (pflag && (plines ) 54))       /* If pflag set and number of lines ) 54. */
        {
           plines = 0;                    /* Initialize plines. */
           lprtff();                      /* Print line. */
           for (i=1; i (= 6; i++) lprtlf(); /* Increment i from 1 to 6. */
        }
        if (pflag)                        /* If pflag set. */
        {
           if (col ) 1) for (i=1; i ( col; i++) lprtchar(0, ' ');
           lprtstr (line);                /* Print string. */
           lprtcr();
           lprtlf();                      /* Print line. */
           plines++;                      /* Increment plines. */
        }
        curlocat (++row, col);            /* Start writing with cursor at (row,col). */
        colrprts (line, FOREGRND, BACKGRND); /* Clear screen and set colors. */
     }
   }

   fclose (infile);                       /* Close file buffer. */
   if (pflag)                             /* If pflag set. */
   lprtff();                              /* Print infile. */
   return;
}
```

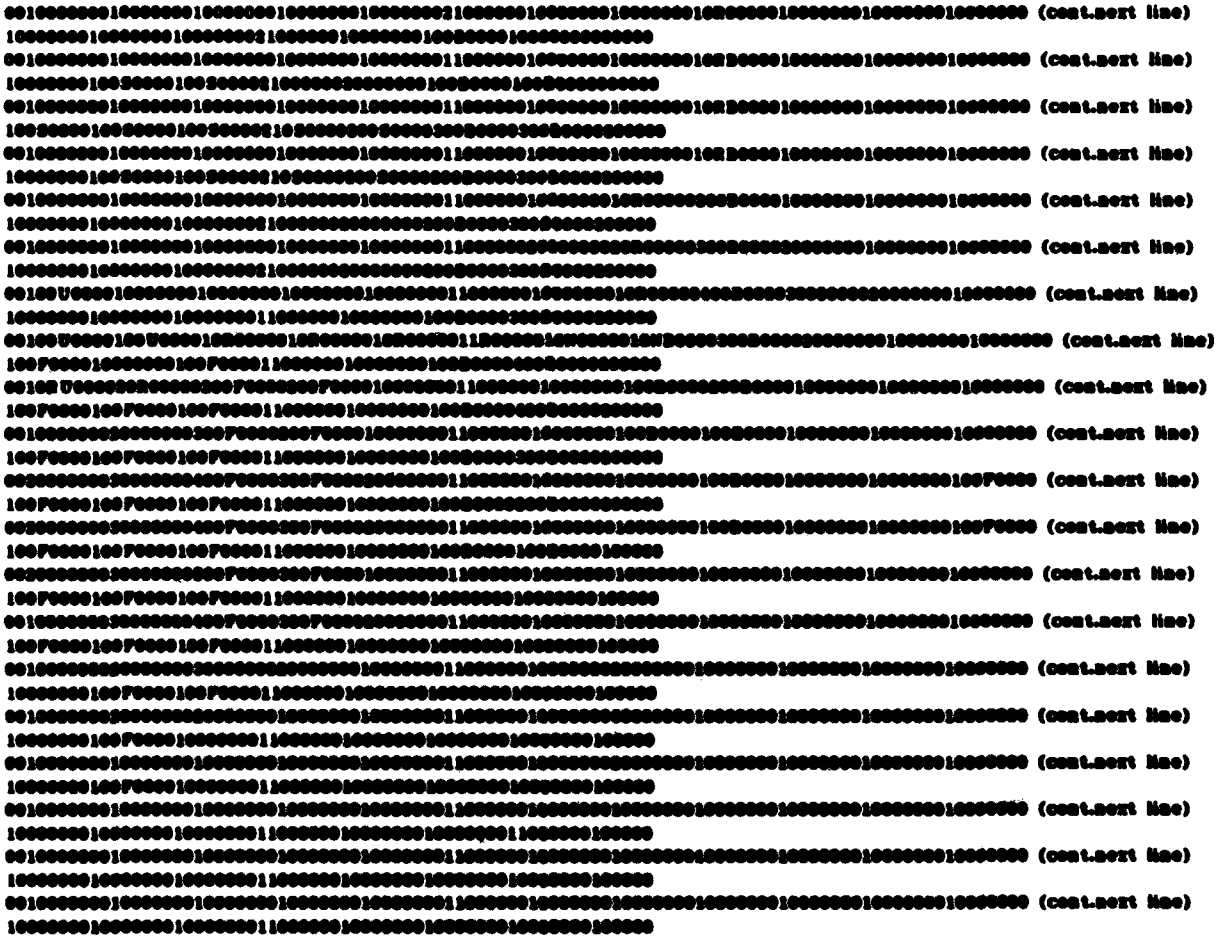
```
/*************************************************************************
*                                                                        *
*                              TIMELOOP                                  *
*                                                                        *
*************************************************************************/

/*****
  Timeloop displays the system time and date at the bottom of the menu title board.

PARAMETERS:
  None.
*****/

timeloop()

{
   int i, j, row, col;                    /* Local integer variables. */
   char dt[30],                           /* Date display buffer. */
        tm[12];                           /* Time display buffer. */

   clearkbd();                            /* Clear keyboard buffer. */
   curget(&row, &col); /* Place cursor at location of pointers (row,col). */
   for( ; ; )                             /* Start infinite loop. */

   {
      curtype (1, 0, 0);                  /* Turn cursor off. */
      if((i = checkkey()) == 1) break;    /* Wait for key pressed. */
      systime (&tm, 9);                   /* Load the systime buffer in the #9 format. */
      sysdate (&dt, 24);                  /* Load the sysdate buffer in the #24 format. */
      curlocat ( 8,18);                   /* Start writing at cursor location (8,12). */
      colrprts(dt,FOREGRND,BACKGRND);     /* Print contents of the sysdate buffer in set colors. */
      curlocat( 8,52);                    /* Start writing at cursor location (8,46). */
      colrprts (tm, FOREGRND, BACKGRND);  /* Print contents of the systime buffer in set colors. */
      curlocat(row,col);                  /* Start writing at cursor location (row,col). */
      curtype(0,6,7);                     /* Turn cursor on and set colors. */
      for (i = 1; i <= 1000; i++)         /* Increment i from 1 to 1000. */

      {
         if((j = checkkey()) == 1) goto done; /* On key pressed log off and exit to DOS. */
      }
   }

   done:  curtype(0,6,7);                 /* Turn cursor on and set colors. */
   return;
}


/*************************************************************************
*                                                                        *
*                              ERROR                                     *
*                                                                        *
*************************************************************************/

/*****
  The error routine beeps the speaker and returns the cursor to the same column, previous row.
  It is used to signal bad data input.

PARAMETERS:
  None.
*****/

error()

{
   int row, col;                          /* Initialize row, col integer variables. */
   curget (&row, &col);                   /* Set location of cursor at pointers row, col. */
   curlocat (row, col - 1);               /* Start writing at cutsor location (9,2). */
   beep();                                /* Sound the error warning. */
```

```
   return;
}


/*************************************************************************
*                                                                        *
*                              ERRMSG                                    *
*                                                                        *
*************************************************************************/

/*****
  This routine prints a centered error message on lineno, beeps once and pauses.
  The user strikes a key, the message is cleared and control is returned.

PARAMETERS:
  lineno -- Location of the cursor and the error message.
  msg    --  Error message to be printed.
*****/

errmsg (lineno, msg)

int lineno;                             /* External integer variable. */
char *msg;                              /* Pointer to external character variable. */

{
   int i, mode;                         /* Local integer variables. */
   getscmod (&mode, &i, &i);            /* Determine screen display mode and set colors. */
   if (mode) i = (80 - strlen (msg)) /2; /* Calculate center of string in 80 column mode. */
   else i = (40 - strlen (msg)) /2;     /* Calculate center of string in 40 column mode. */
   curlocat(lineno, i);                 /* Start writing at cursor location (9,2). */
   colrprts(msg, ERRCOLOR,BACKGRND);    /* Print in set colors. */
   beep();                              /* Sound error warning. */
   getkey (&i);                         /* Wait for key pressed. */
   curlocat(lineno, i);                 /* Start writing at cursor location (9,2). */
   colrprts(
                        FOREGRND,BACKGRND);    /* Print in set colors. */
}



/*************************************************************************
*                                                                        *
*                              QUIT                                      *
*                                                                        *
*************************************************************************/

/*****
  Quit causes the program to  exit to DOS.

PARAMETERS:
  None.

MENU:


             YOU ARE ABOUT TO EXIT TO DOS !!!



                 Quit Y/N? ___

*****/
```

```
quit()

{
   int i;                                /* Local integer variables. */

   clscolor (FOREGRND, BACKGRND);        /* Clear screen and set colors. */
   clearkbd();                           /* Clear keyboard buffer. */
   page ("QUIT.MNU");                    /* Display the Exit Menu. */
   curlocat (24, 30);                    /* Set cursor at location (24,30). */


      i= ecoyesno(24, 30, 1);            /* Check keyboard for a Y/N input with echo. */
      if (i != 1)
      {
         initgraf (0, 0, 0);             /* Clear screen. */
         exit();                         /* Exit to DOS. */
      }
      else
      return;                            /* Return to  routine. */
}


/*****************************************************************************
*                                                                            *
*                                NOTYET                                      *
*                                                                            *
*****************************************************************************/

/*****
  Notyet is a dummy routine to hold a place for future functions.

PARAMETERS:
  None.

MENU:



               THIS FUNCTION NOT YET IMPLEMENTED




      Press a key to continue.

*****/

notyet()

{
   int i;                                /* Local integer variable. */
   clscolor(FOREGRND,BACKGRND);          /* Clear screen and set colors. */
   page ("NOTYET.MNU");                  /* Display the notyet message. */
   curtype (0, 0, 0);                    /* Turn off cursor. */
   curlocat (24, 1);                     /* Set cursor at location (24,1). */
   getkey(&i);                           /* Wait for key pressed. */
   curtype (0, 0, 0);                    /* Turn on cursor. */
   return;                               /* Return to sysmain routine. */
}
```

APPENDIX H

TDMS DATA SET D6009.DAT



REFERENCES

BOOKS AND MANUALS

Field Manual FM 21-32, Topographic Support, Department of the Army, (1979).

Defense Mapping Agency: Digitizing the Future, Defense Mapping Agency, (1987).

Shneiderman, Ben, Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison Wesley, New York, (1987).

Options and Adapters: Volume 2, Technical Reference, International Business Machines, (1984).

MAGAZINES

Alsop, Stewart, "The Enhanced Graphics Standard Comes of Age," PC Magazine, Volume 5 Number 14 (August 1986), 141-143.

Knorr, Eric, ed.,"PC World Graphics Forum", PC World Magazine, Volume 5 Number 2 (February 1987), 210-225

Cooke, Donnald F., "Map Storage on CD-ROM," Byte Magazine, Volume 12 Number 8 (July 1987), 129-138.

Cummings, Steve, "Mastering Corporate Data," PC World Magazine, Volume 5 Number 4 (April 1987), 232-235.

Helliwell, John, "Optical Overview: What's Coming in CD-ROMs and WORMs," PC Magazine, Volume 5 Number 17 (14 October 1986), 149-154.

Howard, Bill, and William G. Wong, "Compaq Leads the Way to Speed and Compatibility," PC Magazine, Volume 5 Number 20 (November 25, 1986), 134-145.

Jadrnicek, Rik, "IBM's Professional Graphics System," Byte Magazine, Volume 10 Number 12 (November 1985), 355-359.

Laub, Leonard, "The Evolution of Mass Storage," Byte Magazine, Volume 11 Number 5 (May 1986), 161-172.

Nichols, Bill, "Inside the 82786 Graphics Chip," Byte Magazine, Volume 12 Number 9 (August 1987), 135-141.

"Optical-Card Reader Uses New Technology", PC Week Magazine, (March 31, 1987), 22.

Rosch, Winn L., "DC2000 Systems: Pocket-Size Backup," PC Magazine, Volume 6 Number 12 (June 23, 1987), 111-133.

Rosch, Winn L., "WORMs for Mass Storage," PC Magazine, Volume 6 Number 12 (June 23, 1987), 135-165.

Somerson, Paul, "How TO Handle Your Hard Disk," PC Magazine, Volume 6 Number 11 (June 9, 1987), 199-229.

Trask, Matt, "386/ASM/LINK 1.1e," Byte Magazine, Volume 12 Number 9 (August 1987), 224-227.

Wiswell, Phil, Vincent Puglia, and Philip F. H. Rose, "Behind the Screens: EGA and Multiscan Monitors," PC Magazine, Volume 6 Number 6 (March 31, 1987), 107-143.

UNPUBLISHED PAPERS

Barnes, Robert B. and Frank J. Sukernick, The Optical Memory Card: Its Role in Distributing Digital Information from Large Databases, Drexler Technology Corporation, (1986). (Presented to SPIE -- The International Society for Optical Engineering at its 30th Annual International Technical Symposium on Optical and Optoelectronic Applied Sciences and Engineering.)

Bruce, S., Fort Lewis DMA Topology Conversion to POINT.DAT, Electronic Data Systems, (1985).

Guth, Peter L., Eugene K. Ressler, and Todd S. Bacastow, MICRODEM: Microcomputer Program For Manipulating Large Digital Terrain Models, U.S. Military Academy, West Point, NY, (1985).

Guth, Peter L., TERRANAL: Microcomputer Terrain Mapping Package, University Of Nevada, Las Vegas, NV, (1986). (Submitted for March 1986 meeting, American Congress on Surveying and Mapping.)

Hoffman, D., D. H. McCoy and H.J. de St. Germain, VISTA (Vision, Intervisibility, Surrogate Travel, Terrain Analysis, Analysis), U.S. Army TRADOC Systems Analysis Activity, White Sands, NM.

BIBLIOGRAPHY

BOOKS AND MANUALS

Field Manual FM 5-36, Route Reconnaissance and Classification, Department of the Army, (1985).

Field Manual FM 30-10, Military Geographic Intelligence (Terrain), Department of the Army, (1972).

Field Manual FM 21-26, Map Reading, Department of the Army, (1969).

Field Manual FM 21-33, Terrain Analysis, Department of the Army, (1978).

Site Data User Guide, Amphibious Objective Area Land Management System, Naval Civil Engineering Laboratory, (1985).

Student Handout DMS No. ST 330, Procedural Guide for Preparation of Cross-Country Movement (CCM) Overlays, Defense Mapping School, (1984).

MAGAZINES

Antonuccio, Antony, "Tape Backup Systems," Byte Magazine, Volume 11 Number 5 (May 1986), 227-232.

Byers, T. J., "Built by Association," PC World Magazine, Volume 5 Number 4 (April 1987), 244-251.

Getts, Judy, "A PC Genealogy," PC World Magazine, Volume 5 Number 8 (August 1987), 200-205.

Luhn, Robert, "PC WOrld CD ROM Forum,"PC World Magazine, Volume 5 Number 4 (April 1987), 220-231.

Malloy, Rich, "A Roundup of Optical Disk Drives," Byte Magazine, Volume 11 Number 5 (May 1986), 215-224.

Nihei, Wes, "CD ROM Resource Guide," PC World Magazine, Volume 5 Number 4 (April 1987), 256-259.

Rosch, Winn L., "Backup Choices from Tapes to Disks to WORMS," PC Magazine, Volume 6 Number 12 (June 23, 1987), 101-108.

Zoellick, Bill, "CD-ROM Software Development," Byte Magazine, Volume 11 Number 5 (May 1986), 177-188.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Camron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Dr. Peter Guth Dept. GeoScience University of Nevada Las Vegas, NV 89154	1
4. Mr. D. Hue McCoy US Army TRADOC Systems Analysis Activity ATTN: ATOR-TFC White Sands Missile Range New Mexico 88002-5502	1
5. Dr. Carter Ward Code L64 Naval Civil Engineering Laboratory Port Hueneme, CA 93043	1
6. Capt. Michael R. Reading DMS/MTA Ft. Belvoir, VA 22060	1
7. Capt. Gene Ressler Dept. of Geography & Computer Science United States Military Academy West Point, NY 10996-1695	1
8. Capt. D. A. Frakes PM PLRS, CECOM Ft. Monmouth, NJ 07703	1
9. Cdr. J. S. Stewart II Code 55 ST Naval Postgraduate School Monterey, CA 93943	2
10. Chief Of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, DC 20350-2000	1

END

DATE

FILMED

DEC.

1987